AD-A241 900

~~AD-A238 705~~

REPORT DOCUME

1a. REPORT SECURITY CLASSIFICATION: Unclassified	
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFOSR-90-0224	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 91 [illegible]
6a. NAME OF PERFORMING ORGANIZATION: Yale University; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: Same as 8a
6c. ADDRESS (City, State and ZIP Code): 333 Cedar Street, New Haven, CT 06510	7b. ADDRESS (City, State and ZIP Code): Same as 8C
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-90-0224
8c. ADDRESS (City, State and ZIP Code): Bolling AFB, Washington, DC 20332 — Bldg 410	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2305	B3	

11. TITLE (Include Security Classification): Neural Networks for Model-based Recognition

12. PERSONAL AUTHOR(S): Gene R. Gindi

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Annual	FROM 5/1/90 TO 5/1/91	6/12/91	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This annual progress report describes first-year progress in theoretical and applied fronts for neural-net object recognition via graph matching. On the theory front, a learning scheme is applied to our previously hand-designed graphs, and a Bayesian approach to weighted graph matching is described. On an applied front, our networks are applied to recognition of machined parts. Continuing progress on the application of "continuation" optimization methods to our networks is reported.

91-13786

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: unclass
22a. NAME OF RESPONSIBLE INDIVIDUAL: Suddarth	22b. TELEPHONE NUMBER (Include Area Code): 202-767-4931; 22c. OFFICE SYMBOL: NE

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.   SECURITY CLASSIFICATION OF THIS PAGE

91 10 22 086

First Year Progress Report
AFOSR Grant AFOSR-90-0224
Neural Networks for Model Based Recognition

Gene Gindi
Yale University

√

A-1

Overview

This is a progress report for the first year of a three-year grant. It is organized as follows: The overall nature and scope of the work is first described. We then summarize oriefly progress made in several areas. Each of these areas is then explained in more detail in papers, theses, internal reports that follow this brief overview. These latter documents are self sufficient in containing the necessary background, introductions and references.
This grant involves continuing work in the a neural-net methodology for high-level vision. We use an optimization approach and optimizing neural networks to perform the sorts of combinatorial searchs that arise inevitably in high-level vision. The premise is that high-level vision is in essence a *matching* process in which complex models are matched to complex objects. By complex, we imply objects that are composed of many parts and have many degrees of freedom. We are working on sev for these nonconvex problems (3) trying this out on real objects (4) a firmer theoretical framework based on Bayesian principles.
Below we describe progress in three areas:

- A learning algorithm was proposed and implemented for our graph-matching networks. Previously, models of objects were stored in the network by a simple hand-design method. The new method allows for a supervised learning methodology for storing the models. The actual networks use a combination of backprop and clustering to approximate curves that tend to attain significant values in localized regions (i.e. learning bumps) The approach and experiments are described in a completed thesis by Grant Shumaker, a copy of which is enclosed. This thesis also describes numerical simulations in object recognition with an earlier proposed "Lagrange multiplier" network for constrained optimization.

- To date, our networks have been applied to artificial object domains and our focus has been on theory and methodology. In a paper by Volker Tresp (this will appear in NIPS-3), results in the recognition of real objects (machined parts) are reported. Significant here too is the use of a continuation method (two types of mean-field annealing) that leads to much improved convergence and ability to escape local minima. In one experiment, Tresp improves on a method proposed by Peterson and Soderberg (IJNS vol. 1 1989) by incorporating more complex constraints into the optimization method.

- Our networks perform a sort of weighted graph matching, but their design is heuristic in several respects. In an internal report, Utans and Gindi describe a more pleasing approach in which weighted graph matching exists as a form of Bayesian inference.

Weighted Graph Matching as Bayesian Inference

Joachim Utans and Gene Gindi
Yale University
Department of Electrical Engineering
P.O. Box 2157, Yale Station New Haven, CT 06520

June 18, 1991

Internal Report – December 90

1 Weighted Graph Matching

Consider the following problem of pure graph matching (see Figure 1). Given two graphs each consisting of N nodes, let α $(1 \leq \alpha \leq N)$ index the nodes of one of the graphs, and i $(1 \leq i \leq N)$ index the nodes of the other. Let $G_{\alpha\beta}$ be the adjacency matrix of one of the graphs,

$$G_{\alpha\beta} = \begin{cases} 1 & \text{if } \alpha \text{ and } \beta \text{ are connected by an arc} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Likewise g_{ij} is the adjacency matrix of the other graph. A sparse match matrix $\{M_{\alpha i}\}$, $M_{\alpha i} \in (0, 1)$, of dynamic variables represents the correspondence between nodes α and i.

Graph matching then means to find a match matrix M that maps nodes in $G_{\alpha\beta}$ to nodes in g_{ij} in a structurally consistent way; i.e. finding consistent rectangles (see figure 1). In a weighted graph matching problem, arcs have numerical weights $W_{\alpha\beta}$ and w_{ij} attached; then the optimal mapping also minimizes the difference of the weights of arcs connecting matched nodes.

2 Graph Matching as Bayesian Inference

The problem we are addressing here involves finding a match matrix M given the data graph g_{ij} and the model information (the model graph $G_{\alpha\beta}$). In a Bayesian sense we want to find the MAP estimate for the best M given the data as expressed by Bayes theorem:

$$P(M|g) = \frac{P(g|M)P(M)}{P(g)} \tag{2}$$

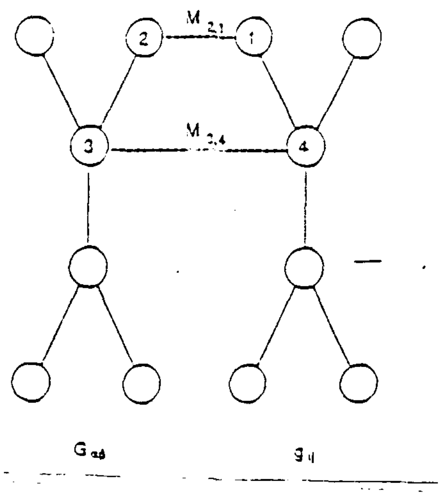


Figure 1: Graph Matching: The circles represent graph nodes, the solid lines graph arcs and the dotted lines matches between nodes in two different graphs. The nodes 1,2,3 and 4 form a consistent rectangle.

Here, $P(M)$ is the prior regarding a valid match matrix M and

$$P(g) = \sum_M P(g|M)P(M) \tag{3}$$

a normalizing constant. Thus, we are left with finding an expression for $P(g|M)$. This can be interpreted as finding the "forward" model (or grammar) that describes the generation of typical data graphs assuming that we are given a valid match matrix M.

We proceed by assuming that arcs in g are generated independently and thus

$$P(g|M) = \prod_{ij} P(g_{ij}|M) \tag{4}$$

This might not be a realistic assumptions for if g decribes an object it is more reasonable to assume that there exists some correlation; for the moment we make this assumption for simplicity.

An arc g_{ij} will be present in the data graph if both, nodes i and j, get mapped to nodes α and β in the model graph. This means that elements $M_{\alpha i}$ and $M_{\beta j}$ of M are both 1. In addition, nodes α and *beta* must be connected by an arc $G_{\alpha\beta}$. Thus

$$P(g_{ij} = 1|M) = M_{\alpha i} M_{\beta j} G_{\alpha\beta} \tag{5}$$

where $M_{\alpha i}$ and $M_{\beta j}$ are used as "switch" terms. Since g_{ij} can assume only two values,

$$P(g_{ij} = 0|M) = 1 - P(g_{ij} = 1|M) \tag{6}$$

$G_{\alpha\beta}$ in the mode graph can be given a probabilistic meaning: it is the probability that in a typical instance g of G a particular arc is present.

For the match between model and data to be unique, M must be a permutation matrix: $M_{ij} \in [0,1]$ and $\sum_i M_{ij} = 1, \sum_j M_{ij} = 1$. If M is chosen from the domain of permutation matrices, $P(M)$ would just be a constant (assuming uniform distribution). Here, we want to implement the elements of M as independent match neurons and therefore have to specify the prior distribution in such a way that valid permutation matrices are more likely than others:

$$P(M) = e^{-A\sum_j(\sum_i M_{ij}-1)^2 - B\sum_i(\sum_j M_{ij}-1)^2} \tag{7}$$

which is a Gaussian approximation to the ideal distribution

$$P(M) = \prod_{ij} \delta^K(\sum_i M_{ij}, 1)\delta^K(\sum_j M_{ij}, 1) \tag{8}$$

Now, the final distribution is given by

$$P(M|g) \propto \prod_{ij} M_{\alpha i} M_{\beta j} G_{\alpha\beta} e^{-A\sum_j(\sum_i M_{ij}-1)^2 - B\sum_i(\sum_j M_{ij}-1)^2} \tag{9}$$

3 Extension to Weighted Graph Matching

Here, the arcs in the data graph g_{ij} have weights w_{ij} attached; in the "forward" sense, we therefore want to compute

$$P(w_{ij}|M) = \sum_{\{g_{ij}\}} P(w_{ij}|g_{ij}, M)P(g_{ij}|M) \tag{10}$$

where the summation is taken over all values g_{ij} can assume, i.e. 0 and 1. Next we define

$$P(w_{ij}|g_{ij}, M) = \begin{cases} \left(\frac{1}{\sqrt{2\pi}\sigma_w}\right) e^{-\frac{1}{2\sigma_w^2}(W_{\alpha\beta} - M_{\alpha i}M_{\beta j}w_{ij})^2} & \text{if } g_{ij} = 1 \\ 0 & \text{if } g_{ij} = 0 \end{cases} \tag{11}$$

or, using g_{ij} as a "switch" term

$$P(w_{ij}|g_{ij}, M) = g_{ij}\left(\frac{1}{\sqrt{2\pi}\sigma_w}\right) e^{-\frac{1}{2\sigma_w^2}(W_{\alpha\beta} - M_{\alpha i}M_{\beta j}w_{ij})^2} \tag{12}$$

This expression states that if there is no arc g_{ij} in the data graph then there can be no weight w_{ij}; otherwise, the weight is determined by the model (Gaussian distributed with mean $W_{\alpha\beta}$ and variance σ_w). Thus,

$$P(w_{ij}|M) = g_{ij}\left(\frac{1}{\sqrt{2\pi}\sigma_w}\right) e^{-\frac{1}{2\sigma_w^2}(W_{\alpha\beta} - M_{\alpha i}M_{\beta j}w_{ij})^2} P(g_{ij} = 1|M) \tag{13}$$

which allows us to use the results from the previous section. For weighted graph matching we thus obtain

$$P(M|g,w) \propto \prod_{ij} M_{\alpha i} M_{\beta j} G_{\alpha\beta} e^{-A\sum_j(\sum_i M_{ij}-1)^2 - B\sum_i(\sum_j M_{ij}-1)^2} \quad (14)$$
$$g_{ij}\left(\frac{1}{\sqrt{2\pi}\sigma_w}\right) e^{-\frac{1}{2\sigma_w^2}(W_{\alpha\beta}-M_{\alpha i}M_{\beta j}w_{ij})^2}$$

Acknowledgements

We benefitted from helpful discussions with E. Mjolsness.

to appear in NIPS →

A Neural Network Approach for Three-Dimensional Object Recognition

Volker Tresp
Siemens AG, Central Research and Development
ZFE IS INF2, Otto-Hahn-Ring 6, 8000 München 83

Abstract

This poster presents a model-based neural vision system. Scenes are described in terms of shape primitives and their relational structure. The neural network matches the primitives in the scene and the primitives in a model base by finding the best agreement between primitives and their relational structure under the constraint that at most one primitive in the model base should be assigned to a primitive in the scene. The quality of the solutions and the convergence speed were both improved by using mean field approximations. The approach was tested in 2-D and in 3-D object recognition. The primitives are line segments derived from edges in the scenes. In the 2-D problem, the recognition is independent of position, orientation, size and slight perspective distortions of the objects. In the 3-D problem, stereo images are used to generate a 3-D description of the visible (not occluded) portions of the objects in the scenes. The scene description is matched against objects in a model base. The best match identifies the correct models and using the information obtained in the preprocessing step, the 3-D positions of the objects can be determined.

1 Introduction

Many machine vision systems and, to a large extent, also the human visual system are model based. The scene is described in terms of shape primitives and their relational structure and the vision system tries to find a match between the scene description and 'familiar' objects in a model base. If the scenes are acquired with a single camera and the objects are flat or always viewed from the same perspective, the problem is essentially 2-D. It is often required that the recognition be invariant to rotation, translation, scale and slight perspective distortions. This can be achieved if solely parameters invariant to those transformations are used in the recognition system. In this paper, a system is described in which shape primitives are line segments derived from the edges in the image. The scene is described in terms of the relations of line segments, e.g. the angles between line segments and the logarithms of the ratios of their lengths. In the approach

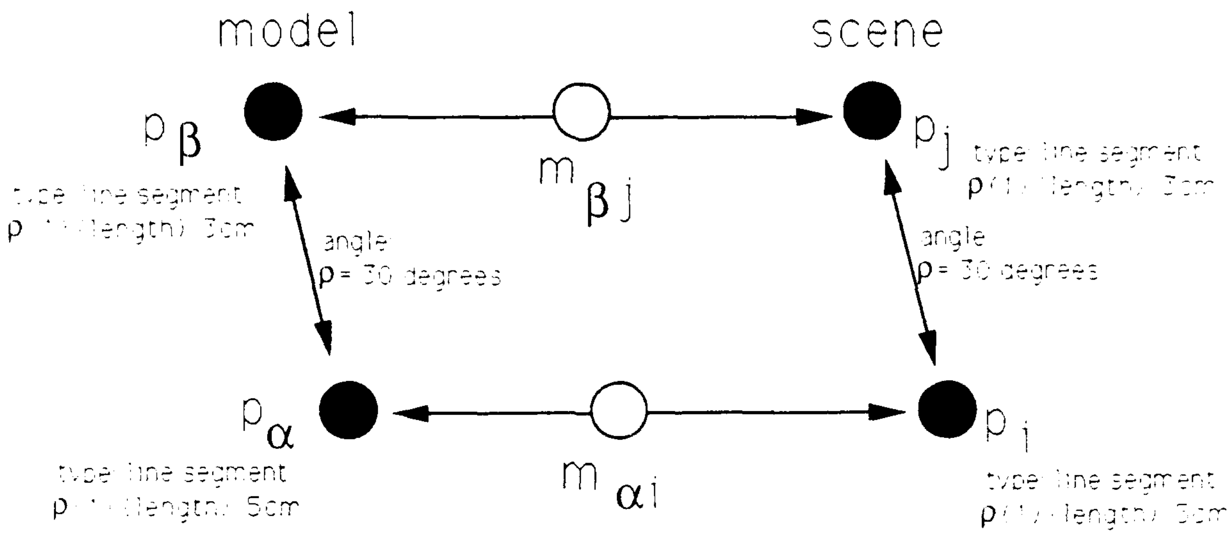


Figure 1: Match of primitive p_α to p_i.

presented here, a neuron is assigned to every possible match between primitives in the scene and the model base. The network is designed to find the best match between the scene description and the model base under the constraint that at most one primitive in the model base is assigned to a primitive in the scene description.

If the problem is intrinsically 3-D as in many robotics applications, the vision system should capture the true 3-D structure of the scene. Using the sensory information available, a 3-D description of the scene must be generated which can then be compared to 3-D descriptions of models in the model base. In this approach, this information is gained from two stereo-images and the correspondence problem must be solved first. Only part of an object is visible at any one time which in general will only permit a 3-D description of that portion of the object. In this poster, we describe a neural network approach that offers an elegant method to handle the uncertainty in the 3-D scene description and solves both the correspondence problem and the model matching task.

2 The Network Architecture

The activity of a match neuron $m_{\alpha i}$ represents the certainty of a match between a primitive p_α and in the model base and p_i in the scene description. The connectivity of the network is most easily described by the network's energy function where the fixed points of the network correspond to the minima of the energy function. The

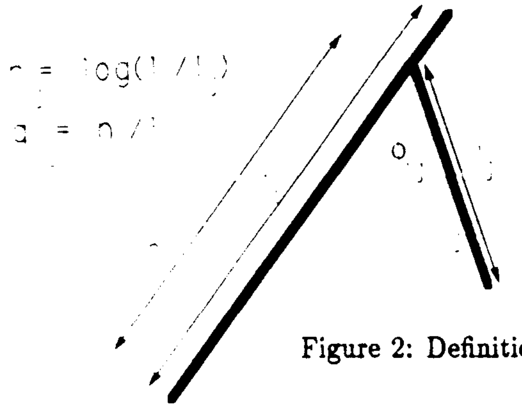


Figure 2: Definitions of r, q, and θ

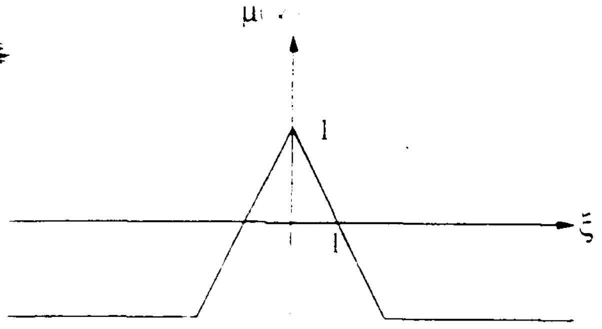


Figure 3: The function $\mu()$.

energy function in the system described here is the sum of several terms. The first term evaluates the match between the primitives

$$E_P = -1/2 \sum_{\alpha i} \kappa_{\alpha i} m_{\alpha i}. \qquad (1)$$

The function $\kappa_{\alpha i}$ is zero if the type of primitive p_α is not equal to the type of primitive p_i. If both types are identical, $\kappa_{\alpha i}$ evaluates the agreement between parameters $\rho^p_\alpha(k)$ and $\rho^p_i(k)$ which describe properties of the primitives. Here, $\kappa_{\alpha i} = \mu(\sum_k |\rho^p_\alpha(k) - \rho^p_i(k)|/\sigma^p_k)$ is maximum if the parameters of p_α and p_i match (Figure 1 3).

A direct comparison of the primitives is not sufficient. The evaluation of the match between the relations of primitives in scene and data base is performed by the energy term [3]

$$E_S = -1/2 \sum_{\alpha,\beta,i,j} \chi_{\alpha,\beta,i,j}\, m_{\alpha i} m_{\beta j}. \qquad (2)$$

The function $\chi_{\alpha i} = \mu(\sum_k |\rho^r_{\alpha,\beta}(k) - \rho^r_{i,j}(k)|/\sigma^r_k)$ is maximum if the relation between p_α and p_β matches the relation between p_i and p_j.

The primitives can be interpreted as nodes in a graph and the relations between the primitives as labeled arcs. Seen in this way, the network solves a graph matching problem [1, 7, 6, 8].

Depending on the application, uniqueness constraints may have to be satisfied. These can be incorporated as additional (penalty-) energy terms. For example, the constraint that a primitive in the scene should only match to one or no primitive in the model base (column constraint) can be implemented by [6]

$$E_C = \sum_i [((\sum_\alpha m_{\alpha i}) - 1)^2 \sum_\alpha m_{\alpha i}]. \tag{3}$$

E_C is equal to zero only if in all columns the sum over the activations of all neurons is equal to one or zero and positive otherwise.

If neurons are employed that can take on continuous values ($m_{\alpha i} \in (0,1)$), an additional term is helpful that encourages neurons to assume values close to zero or one

$$E_B = \sum_{\alpha i} m_{\alpha i}(1 - m_{\alpha i}). \tag{4}$$

2.1 Dynamic Equations and Mean Field Theory

2.1.1 MFA_1

The neural network should make binary decisions, match or no match, but binary recurrent networks get easily stuck in local minima. A higher probability of reaching a lower local minimum can be obtained by using the mean field approximation of statistical physics. Here, the network is interpreted as a system of interacting units. If such a system is in thermal contact with a heat reservoir of temperature T the system will be in a state S with the probability

$$\frac{e^{-E(S)/T}}{Z} \tag{5}$$

where

$$Z = \sum_S e^{-E(S)/T} \tag{6}$$

is the partitioning function of the system and $E(S)$ is the energy of the system in state S.

Such a system minimizes the free energy

$$F = E - T\hat{S} \tag{7}$$

where $\hat{S}$ is the entropy of the system. At $T = 0$ the energy E is minimized. Equation 5 indicates that states with low energy are more likely but since there are more states accessible with a higher energy the system is with a high probability close to the mean energy $\sum_S E(S)e^{-E(S)/T}/Z$.

Bad local minima can be avoided by using an annealing strategy but annealing is time consuming when simulated on a digital computer. Using a mean field approximation one can obtain deterministic equations by retaining some of the advantages of the annealing process [5, 4, 2].

The mean field theory gives a good approximation for a highly interconnected network with many neurons. The theory works under the assumption that the change in energy of the system if one neuron changes its state is approximately equal to the change in energy of that neuron in the field at its location. The mean value $v_{\alpha i} = < m_{\alpha i} >$ of a neuron becomes

$$v_{\alpha i} = \frac{1}{\sum_S e^{-E(S)/T}} \Big(\sum_{S, m_{\alpha i}=1} 1 \times e^{-E(S)/T} + \sum_{S, m_{\alpha i}=0} 0 \times e^{-E(S)/T} \Big) \quad (8)$$

$$\approx \frac{1 \times e^{u_{\alpha i}/T}}{e^{u_{\alpha i}/T} + e^{-0/T}} = \frac{1}{1 + e^{-u_{\alpha i}/T}} \quad (9)$$

with

$$u_{\alpha i} = -\frac{\partial E}{\partial v_{\alpha i}}. \quad (10)$$

Equations 9 and 10 can be updated synchronously, asynchronously or solved iteratively by moving only a small distance from the old value of $u_{\alpha i}$ in the direction of the new mean field

$$u_{\alpha i}(t) = u_{\alpha i}(t-1) + \Delta[-\frac{\partial E}{\partial v_{\alpha i}(t-1)} - u_{\alpha i}(t-1)]. \quad (11)$$

At high temperatures T, the system is in the trivial solution $v_{\alpha i} = 1/2 \ \forall \alpha, i$ and the activations of all neurons are in the linear region of the sigmoid function. The system can be described by linearized equations. The magnitudes of all eigenvalues of the corresponding transfer matrix are less than 1. At a critical temperature T_c the magnitude of at least one of the eigenvalues becomes greater than one and the trivial solution becomes unstable. T_c and favorable weights for the different terms in the energy function can be found by an eigenvalue analysis of the linearized equations [5]. MFA_1 is equivalent to the mean field theory of spin glasses [4].

2.1.2 MFA_2

It is also possible to obtain mean field equations which assure that at every temperature T, the column constraint is met. One considers only states S in which exactly one neuron in every column is equal to one and all others are equal to zero or where all neurons in a column are equal to zero. Under the mean field assumption

$$v_{\alpha i} \approx \frac{1 \times e^{u_{\alpha i}/T}}{1 + \sum_\beta e^{u_{\beta i}/T}} \quad (12)$$

with

$$u_{\alpha i} = -\frac{\partial E}{\partial v_{\alpha i}}. \quad (13)$$

The column constraint term (Equation 3) drops out of the energy function. The high temperature fixed point corresponds to $v_{\alpha i} = 1/(N+1) \ \forall \alpha, i$ where N is the number of rows. MFA_2 is similar to the mean field theory of Potts glasses [5].

3 Applications

3.1 2-D Object Recognition

There are many vision applications in which one spatial dimension can be neglected and the object recognition problem is essential 2-D. Figure 4, 6 and 7 shows typical objects presented to the recognition system. The task was a recognition of the objects in the scenes independent of scale, translation, rotation and small distortions. The primitives are line segments that approximate edges in the scene. As preprocessing steps, the images are threshholded to separate objects from background. A convolution by a Laplacian followed by another threshholding operation extracts the edges. Edges are found typically at the outer contours of an object or at features inside an object, typically at holes (inside contours). A contour is traced with a simple contour following scheme. The angle ψ which the contour forms in relation to a horizontal line is plotted versus the arclength of the boundary transversed s. Corners of the object correspond to maxima and minima of the smoothed first derivative $d\psi/ds$ and line segments are formed by connecting two successive corners. Contours found within another outer contour are considered an inner contour of the same object (Figures 5, 6, 7)

A single line segment can be described by position, orientation and length. Since none of these parameters is invariant under the transformations mentioned above, a direct comparison between the parameters of the primitives is not feasible and $E_P = 0$. The description of scene and models is encoded in only the relations between line segments. Here, only relations of line segments within a local neighborhood are considered. $\chi_{\alpha,\beta,i,j}$ is equal to zero if not both

- line segment p_α is attached to line segment p_β and
- line segment p_i is attached to line segment p_j.

Otherwise, $\chi_{\alpha,\beta,i,j} = \mu(|\phi_{\alpha\beta} - \phi_{ij}|/\sigma_\phi^r + |r_{\alpha\beta} - r_{ij}|/\sigma_r^r)$ where ϕ is the angle between line segment and r the logarithm of the ratio of their lengths (Figure 2).

The energy terms E_S, E_C and E_B are weighted by 1, 2, and 0.1, respectively, if MFA_1 was used. If MFA_2 is employed, the energy function consists only of E_S, weighted by 1.

3.1.1 Experiments

The model base consisted of 6 different industrial objects which were typically described by 10 to 30 line segments each. The recognition was tested on scenes with a single object in varying scale, position, illumination and orientation. If the illumination allowed a clear separation between background and object, the preprocessing stage segmented the pieces into line segments in the same way as the corresponding pieces in the model base were segmented with variations on the extracted parameters ϕ and r depending on precisely where corners were found. The recognition of the objects was always successful and all line segments matched correctly within about 20 time steps. The network successfully found the correct match even if the data base consisted of two models where one model differed only in one parameter from the other model.

In one scene, all six pieces were present. A total of 81 line segments were matched correctly in about 20 time steps. When the illumination became less uniform, the separation between background and object was not completely possible with simple thresholding. If the segmentation of a contour of an object was correct, that is the same as in the model base, ϕ and r varied up to 15 degrees and 20% respectively but the line segments were matched correctly demonstrating the distortion insensitivity of the system. If portions of a contour were segmented incorrectly, the line segments in that portion were not matched, but the line segments in the correctly segmented portion of the contour were matched correctly. If the model base consisted of all 6 pieces, the line segments in the incorrectly segmented part of the contour were often matched to line segments in the wrong model. Therefore, with large data bases, the image quality should be sufficient to allow a correct segmentation.

The recognition was tested on partially overlapping pieces. If a sufficient number of line segments in the contour of each piece could be segmented correctly, these line segments could be matched and object recognition was successful here as well (Figures 8).

3.2 3-D Object Recognition

3.2.1 The Correspondence Problem

As before, images are segmented into line segments. In the scene in Figure 9, these lines correspond to the edges, structure and contours of the objects and shadow lines. To solve the correspondence problem, corresponding lines in left and right images have to be identified. A good assumption is that the object in one image is a distortion and shifted version of the object in the other image with approximately the same scale and orientation. Therefore, the lengths l of line segments are compared, $\kappa_{\alpha i} = \mu(|l_\alpha - l_i|/\sigma_l^p)$ and the angles ϕ and attachment points q between adjacent line segments are compared, $\chi_{\alpha i} = \mu(|\phi_{\alpha\beta} - \phi_{ij}|/\sigma_\phi^r + |q_{\alpha\beta} - q_{ij}|/\sigma_q^r)$ (Figure 2).

Here, we have two uniqueness constraints: only at most one neuron should be active in each column or each row. The row constraint is enforced by an energy term equivalent to E_C .

$$E_R = \sum_\alpha [((\sum_i m_{\alpha i}) - 1)^2 \sum_i m_{\alpha i}]. \tag{14}$$

Figure 10 shows the line segments and the matrix of match neurons after 10 iterations. All line segments that are present in both images could be matched. One of the legs of the wardrobe was only segmented in the right image and has no correspondence in the left image. Only MFA_2 was employed for this problem. E_S is weighted by 1, E_R by 10 and E_P by 0.1.

3.2.2 Description of the 3-D Object Structure

Next, a a 3-D description of the visible portion of the object must be generated. As result of the last section, we know which endpoints in the left image correspond to endpoints in the right image. In the experiments, the two cameras were mounted in parallel. If D is the separation of both cameras, f the focal lengths of the cameras,

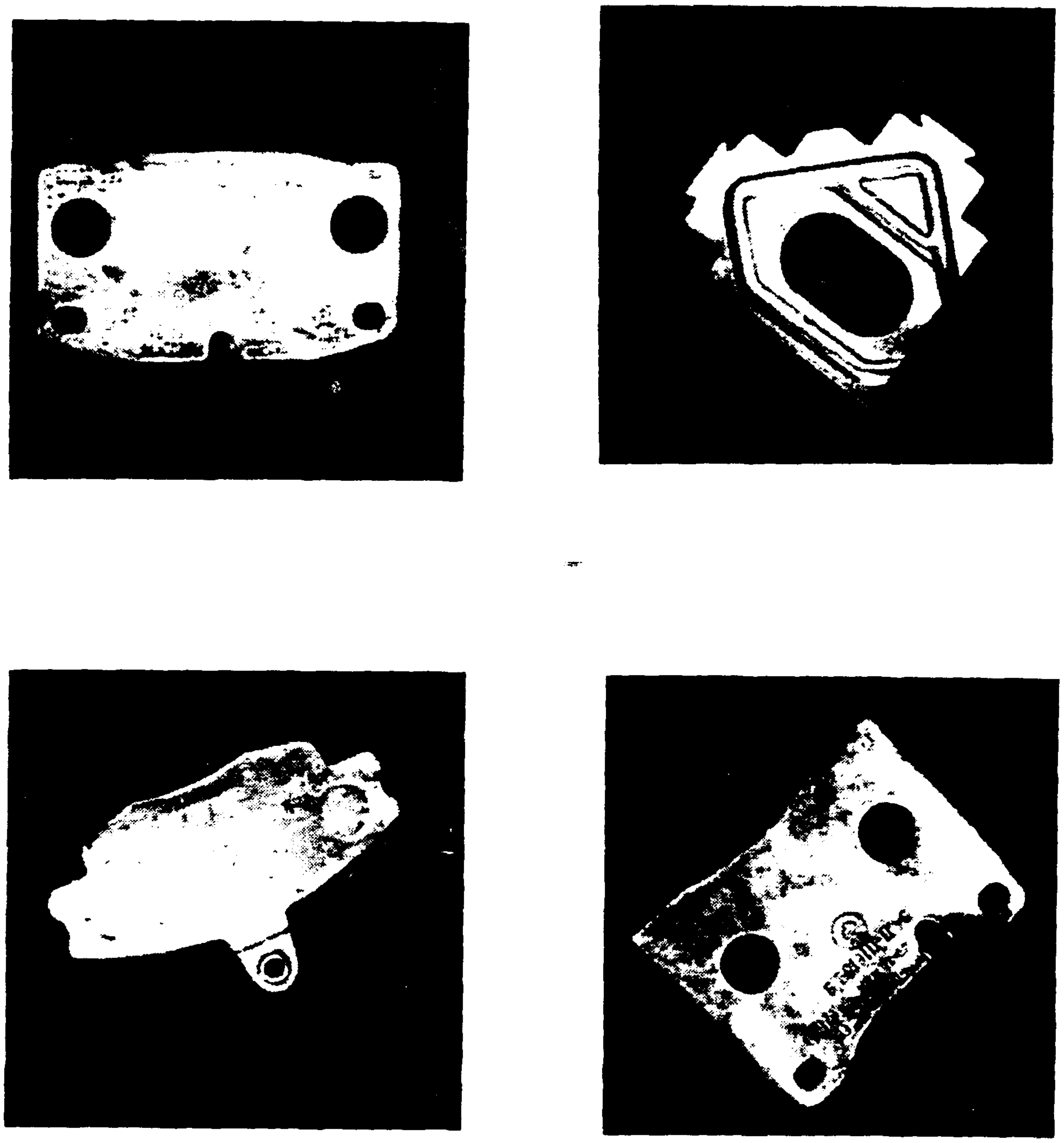


Figure 4: Typical scenes.

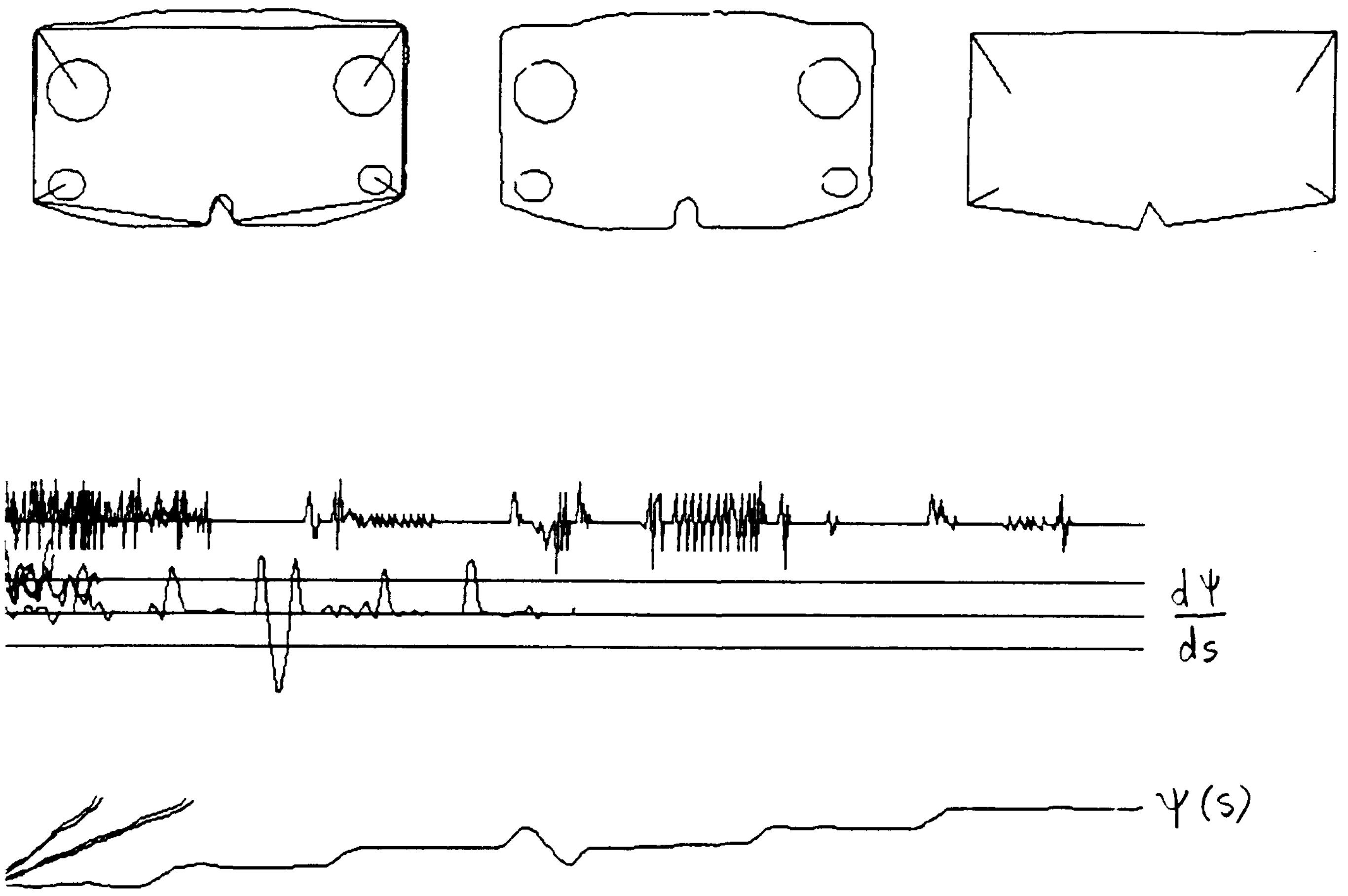


Figure 5: Segmentation.

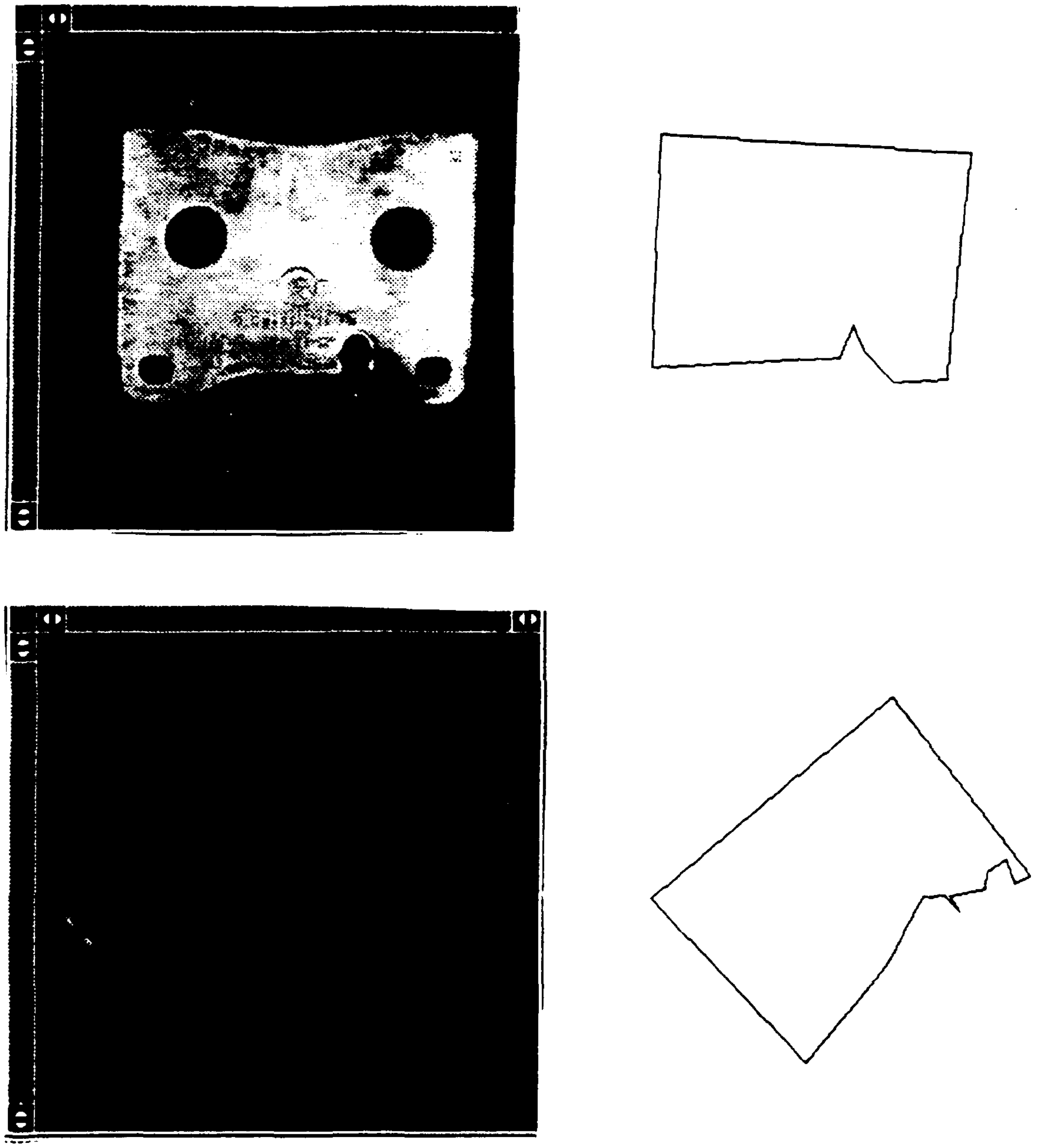


Figure 6: Varying image quality.

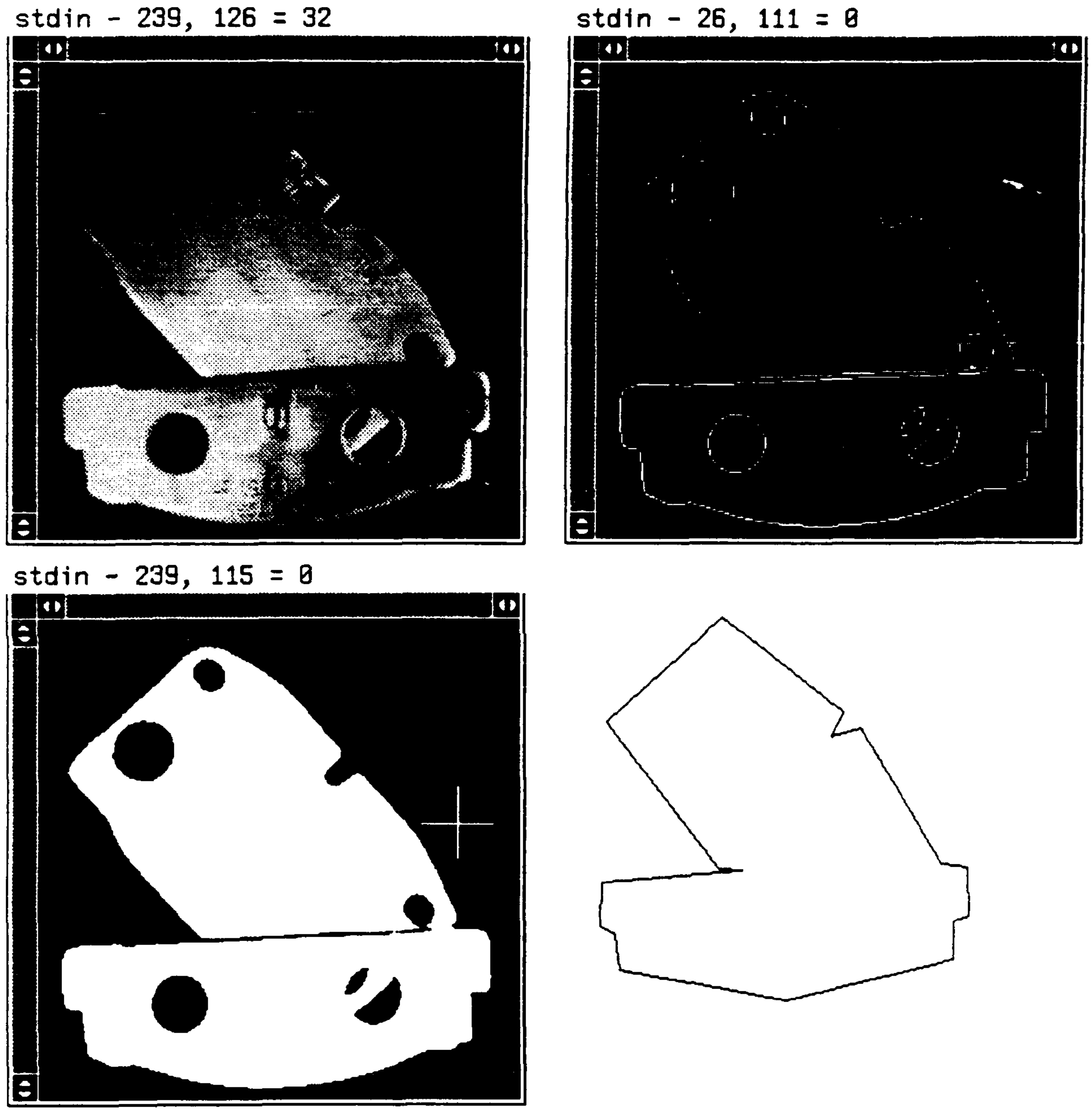


Figure 7: Overlapping workpieces.

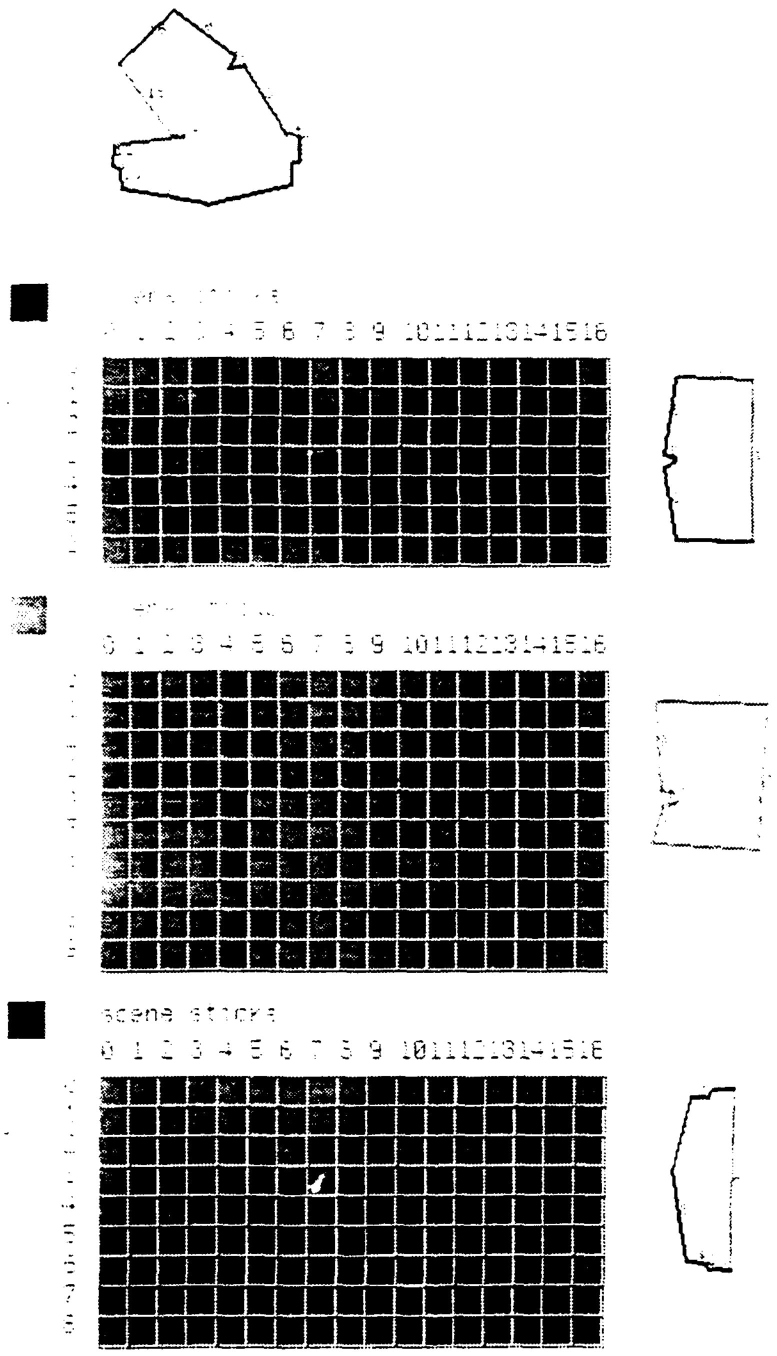
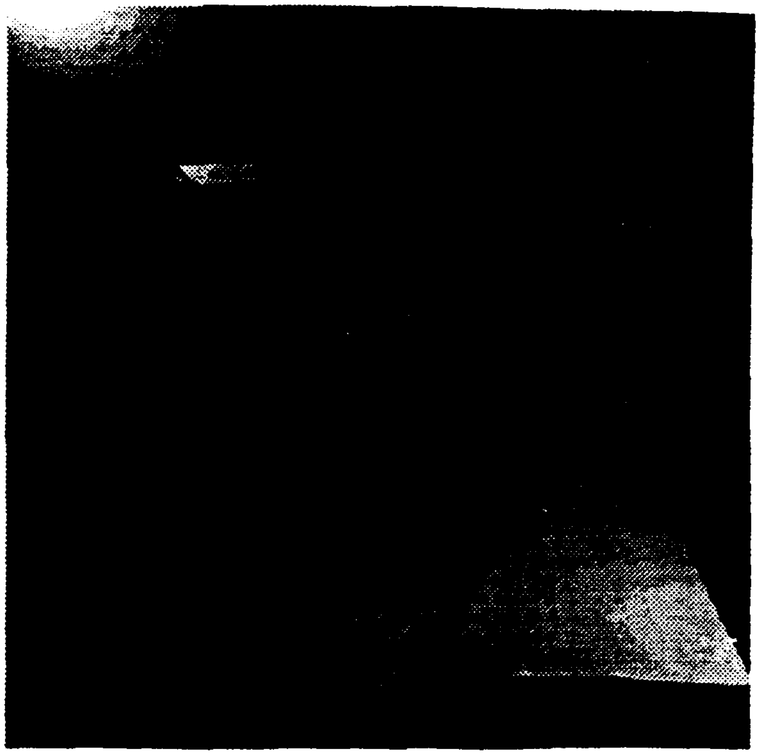


Figure 8: Network converges to solutions. Fat line segments are matched correctly.



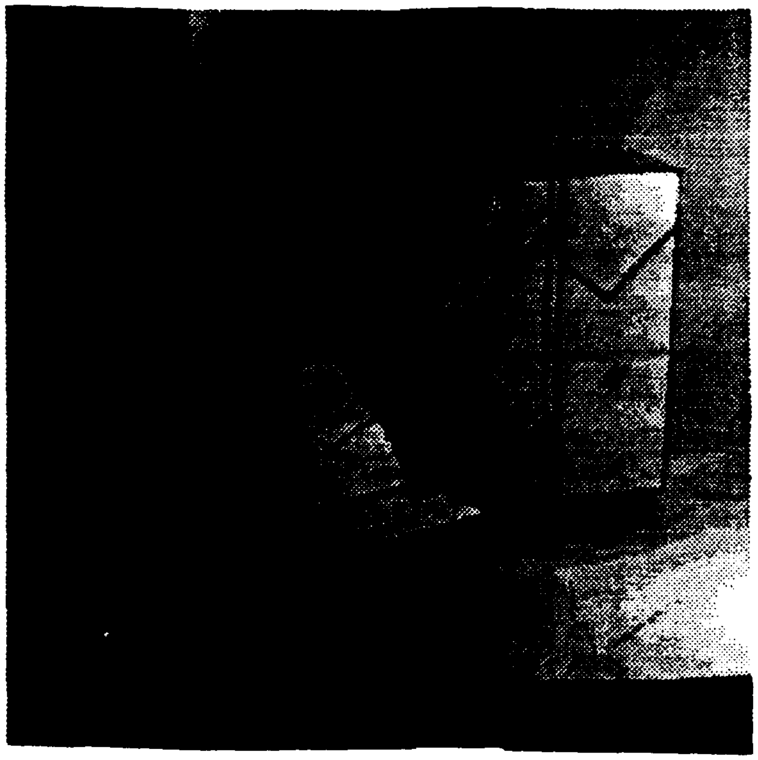


Figure 9: Stereo images of a scene.

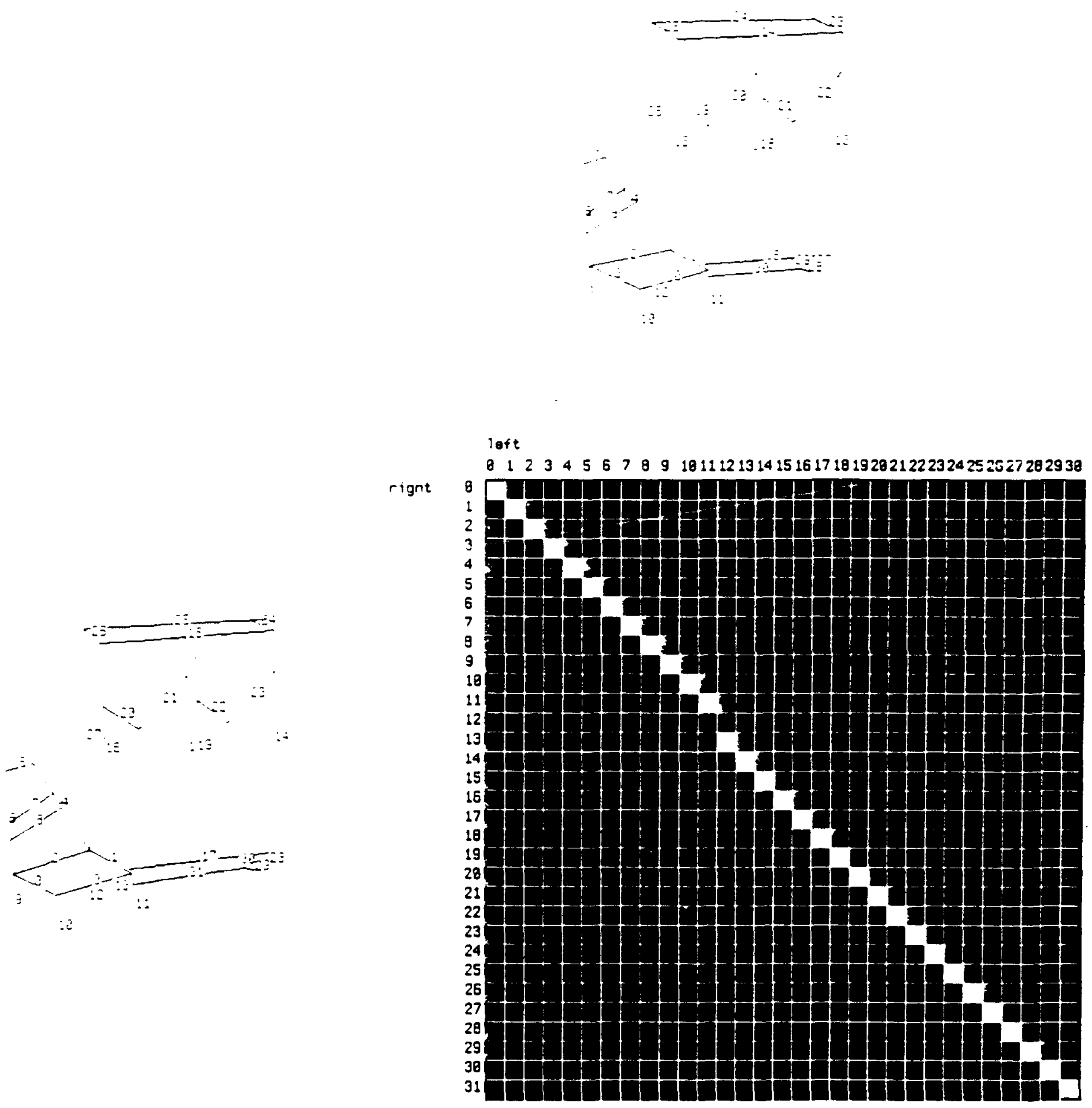


Figure 10: Stereo matching network.

x_r, y_r, x_l, y_l the coordinates of a particular point in left and right images, the 3-D position of the point in camera coordinates x, y, z becomes

$$z = \frac{Df}{x_r - x_l} \tag{15}$$

$$y = zy_r/f \tag{16}$$

$$x = zx_r/f + D/2 \tag{17}$$

Knowing the true 3-D position of the endpoints of the line segments, the system concludes that the chair and the wardrobe are two distinct and spatially separated objects and that line segments 12 and 13 in the right image and 12 in the left image are not connected to either the chair or the wardrobe. On the other hand, it is not obvious that the shadow lines under the wardrobe are not part of the wardrobe.

3.2.3 Matching Objects and Models

The scene description now must be matched with stored models describing the complete 3-D structures of the models in the data base. The model description might be constructed by either explicitly measuring the dimensions of the models or by using several stereo views of the models.

Here, κ and χ are the same as in the correspondence problem. Note, that here θ is not a very discriminating parameter since $\theta \approx 90$ degrees for many pairs of line segments.

The knowledge about the 3-D structure allows a segmentation of the scene into different objects and the row constraint is only applied to neurons relating to the same object O in the scene $E_{R'} = \sum_O \sum_\alpha [((\sum_{i \in O} m_{\alpha i}) - 1)^2 \sum_{i \in_O} v_{\alpha i}]$.

Here, E_S is weighted by 1, $E_{R'}$ by 20 and E_P by 1. Figure 11 shows the network after 20 iterations. Except for the occluded leg, all line segments belonging to the chair could be matched correctly. All not occluded line segments of the wardrobe could be matched correctly except for its left front leg.

4 2-D and 3-D Position

4.1 3-D

A translation followed by a rotation can be described by

$$\begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & dx \\ r_{21} & r_{20} & r_{23} & dy \\ r_{31} & r_{32} & r_{33} & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{pmatrix} \tag{18}$$

or in matrix notation $X_s = RX_0$. The Equation 18 has to hold for all pairs of points $X_0 = (x_0, y_0, z_0)$ in standard position and $X_S = (x_s, y_s, z_s)$ in scene coordinates.

The optimal solution (in the least squares sense) for R is

$$R_{opt} = X_s X_0^P \tag{19}$$

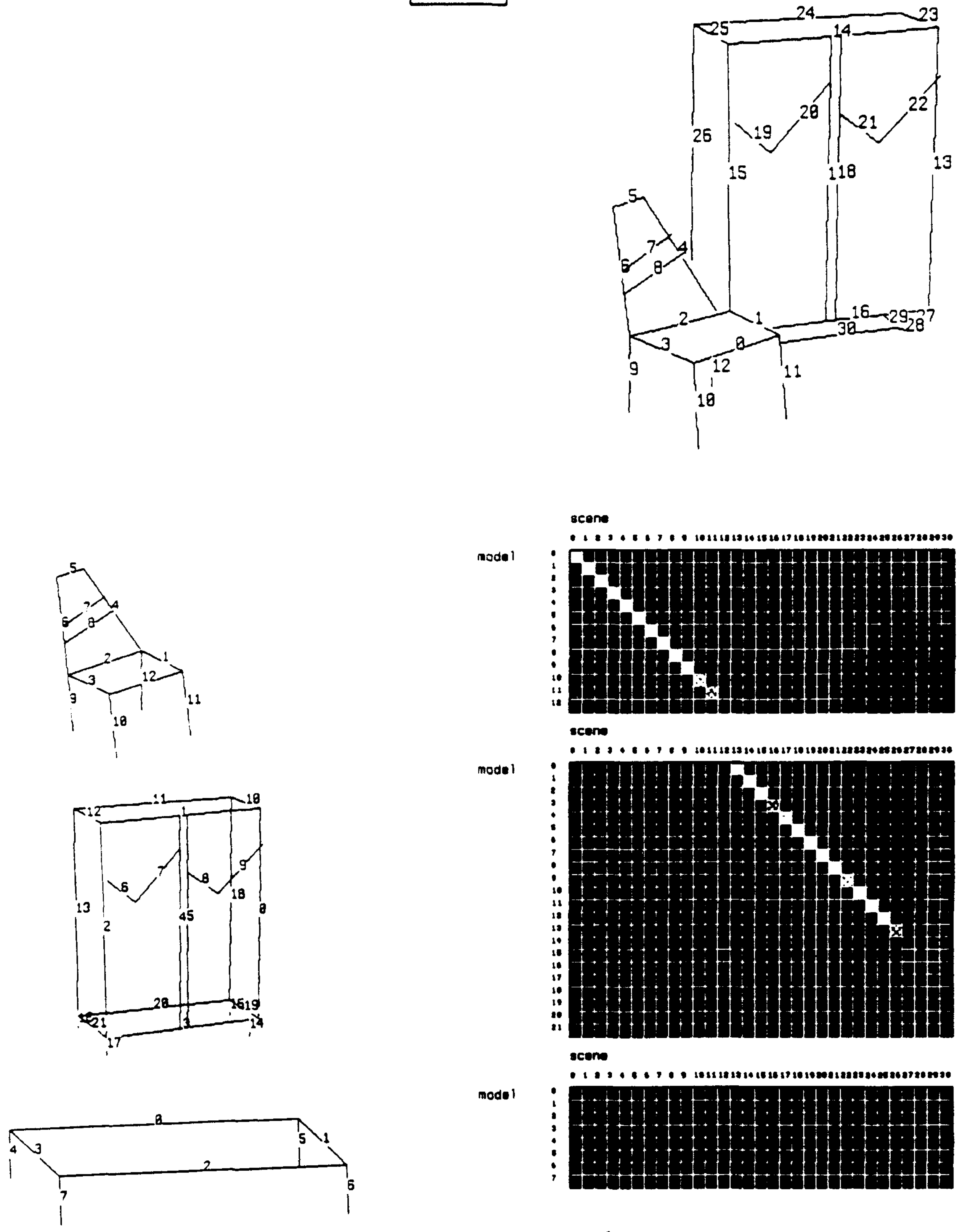


Figure 11: 3-D matching network.

where X_0^P is the pseudo inverse of X_0. The coefficients of R_{opt} can also be obtained using three ADALINEs as shown in Figure 12.

If the first rotation is about the x-axis by γ, followed by a rotation about the y-axis by β, followed by a rotation about the z-axis α

$$\beta = \arctan \frac{-r_{31}}{\sqrt{r_{11}^2 + r_{21}^2}} \tag{20}$$

$$\alpha = \arctan \frac{r_{21}}{r_{11}} \tag{21}$$

$$\gamma = \arctan \frac{r_{32}}{r_{33}} \tag{22}$$

For the chair in Figure 9 one obtains $dx = -3.8$ cm, $dy = 4.8$ cm, $dz = 62.9$ cm, $\gamma = -12$ degrees, $\beta = -36$ degrees and $\alpha = 15$ degrees (in camera coordinates).

4.2 2-D

Scaling, translation and rotation can be described by:

$$\begin{pmatrix} x_s \\ y_s \\ \cancel{z_s} \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda \cos\alpha & -\lambda \sin\alpha & dx \\ \lambda \sin\alpha & \lambda \cos\alpha & dy \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ \cancel{z_0} \\ 1 \end{pmatrix} \tag{23}$$

or in matrix notation

$$X_s = RX_0 \tag{24}$$

The coefficients of R can again be derived using the pseudo inverse or an ADALINE.

5 Convergence and Comparison of the Two Mean Field Approaches

In the 2-D problem, both mean field approaches MFA_1 and MFA_2 were used; in the 3-D problem, only MFA_2 was used. In general, both mean field approaches converged to the same solutions.

If MFA_1 was used, the network was updated using equations 9 and 11. The experiments suggest that there are two critical temperatures: the first when the system leaves the trivial solution $v_{\alpha i} = 1/2 \ \forall \alpha, i$ and starts to enforce the column constraint and the second one when it departs from a solution where all neurons have similar and small activation to the binary solution with only one neuron active per column.

E_C is a penalty term that is introduced to implement a constraint that should be strictly enforced. But unfortunately a large weight C_C for E_C leads to an ill-conditioned Hessian matrix. In practice this means that E_C describes a narrow ravine and the system zig-zags up and down the slopes without making much progress in the perpendicular direction. Also small time steps have to be taken to avoid oscillations

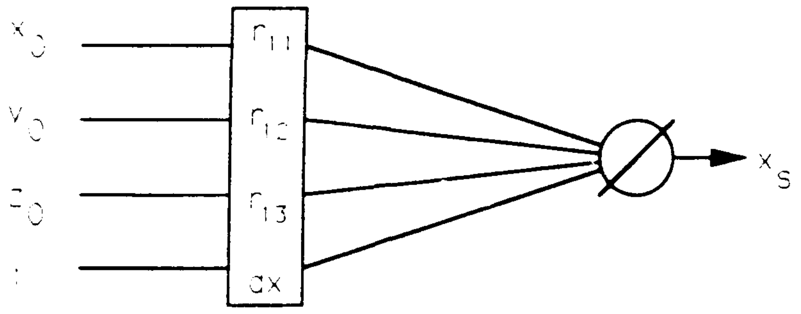


Figure 12: ADALINE for calculation of R.

and instabilities. For these reasons, in some experiments we started with a small C_C and increase it gradually over time. In the experiments, C_C increases exponentially and the step size decreases exponentially. The initial value of C_C is scaled by $1/c$. Through this approach convergence speed could be increased. The network started out slightly above the second critical temperature and annealed from $T = 0.8$ to $T = 0.4$.

When using MFA_2 update equations 12 and 11 were used. Since the energy surface is more favorable, larger time steps can be taken and the net converged much faster ($5 - 10$ times) to a solution without getting into oscillatory states. The annealing started at $T = 0.05$ and was stopped at $T = 0.03$.

6 Discussion

The experiments showed that the system recognizes objects robustly and reliably. The system relies on the correct identification of line segments and their relations in the scene in the preprocessing stage. More elaborate approaches must be used if the scenes become more complex and edges more ambiguous. Edge detection and reliable contour following can be increasingly difficult.

In the 3-D problem, a hierarchical system can be considered. In the first step, simple objects such as squares, rectangles, and circles etc. are identified and these form the primitives in a second stage to recognize complete objects. It is also possible to combine these two matching nets into one hierarchical net as described in [3].

Acknowledgements
I would like to acknowledge the contributions of Gene Gindi, Eric Mjolsness and Joachim Utans of Yale University to the design of the matching network. I thank Peter Kuner and Christian Evers for helping me to acquire the images.

References

[1] E. Bienenstock, C. von der Malsburg, *A Neural Network for Invariant Pattern Recognition*,

Europhys. Lett., 4 (1), 121-126, 1987.

[2] D. E. Van den Bout, T. K. Miller, III *Graph partitioning using annealed neural networks*, IEEE Transactions on Neural Networks, Vol. 1, No. 2 , 1990.

[3] Eric Mjolsness, Gene Gindi, P. Anadan, *Neural Optimization in Model Matching and Perceptual Organization*, Neural Computation 1, 218-209, 1989.

[4] Carsten Peterson, James R. Anderson, *A Mean Field Theory Learning Algorithm for Neural Networks*, Complex Systems 1, 995-1019, 1987.

[5] Carsten Peterson, Bo Soederberg *A new method for mapping optimization problems onto neural networks*, International Journal of Neural Systems, Vol. 1, No. 1, 1989.

[6] Grant Shumaker, Gene Gindi, Eric Mjolsness, P. Anadan, *Stickville: A Neural Net for Object Recognition via Graph Matching*, Tech. Report No. 8908, Yale University, 1989.

[7] Volker Tresp, Gene Gindi. *Invariant Object Recognition by Inexact Subgraph Matching with Applications in Industrial Part Recognition*, Porceedings of the International Neural Network Conference, 1990, Paris, pages 95-98.

[8] Joachim Utans, Gene Gindi, Eric Mjolsness, P. Anadan, *Neural Networks for Object Recognition within Compositional Hierarchies, Initial Experiments*, Tech. Report No. 8903, Yale University, 1989.

Neural Networks for Object Recognition via Graph Matching[*]

A Thesis Submitted to the Yale University School of Medicine in Partial Fulfillment of the Requirements for the Degree of Doctor of Medicine

by

Grant Harper Shumaker

1991

[*]I acknowledge the support of AFSOR, grant F4920-88-C-0025.

Abstract

Neural Networks for Object Recognition via Graph Matching

Grant Harper Shumaker

1991

An objective function for model-based object recognition is formulated and used to specify a neural network whose dynamics carry out the optimization, and hence the recognition task. Models are specified as graphs that capture structural properties of shapes to be recognized. In addition, compositional (*INA*) and specialization (*ISA*) hierarchies imposed on the models as an aid to indexing and are represented in the objective function as sparse matrices. Data are also represented as a graph. The optimization is a graph-matching procedure whose dynamical variables are "neurons" hypothesizing matches between data and model nodes. The dynamics are specified as a third-order Hopfield-style network augmented by hard constraints implemented by "Lagrange multiplier" neurons. Experimental results are shown for recognition in Stickville, a domain of 2-*D* stick figures. *For small databases, the network successfully* recognizes both an object and its specialization. Learning of recognition parameters is implemented through the use of a modified back propagation network.

Contents

List of Figures

List of Tables

1 Vision

1.1 Goals

The process of vision may be formulated as an information-procession task. The visual processing apparatus may be relegated to a isolated black box, with a defined input, in this case a visual scene, and a desired output, the information which is contained within the scene. To formulate the problem in this manner, however, immediately raises fundamental questions about vision. What is the nature of the visual scene used as input? Is it a simple intensity display, does it contain information concerning depth, texture, motion? What is the desired output? Is it simply a catalog of all objects recognized within the object and their positions? This has been a goal of industrial computer vision. Or is the goal to obtain some kind of understanding as to the relationship of objects within the scene; an understanding of the image rather than a processing of the image.

Goals may also be viewed as a hierarchy which emphasize different aspects of the visual process. Object advoidance is important, but may not lead to an understanding of the visual scene. More complex goals may be to extract edge information, characterize objects in the image, or understand their relationship. Additionally, goals at the more complex level may be created by the visual information presented.

A second consideration in vision is in the hardware that is used to implement the visual algorithms within the black box. The nature of an information-processing task suggests, but does not require, particular constraints on the equipment used to process the information. Parallel hardware and algorithms are well suited to processing the vast amounts of information in the initial processes of vision. These constraints direct the search for algorithms used to process the information in the task. Therefore, to understand and characterize any information-processing task, you must study the task as it is accomplished by a particular configuration of equipment used to implement it.

1.2 Psychophysical and Neurophysical studies

One approach to studying visual systems is to disregard what is known about human vision. This approach may indeed have advantages, such as ease of implementation and in industrial applications. However, what is known about the process of visual understanding comes from human vision. It stands as proof that visual understanding is both possible and practical. It provides both a starting point and a working example.

Drawing on previous psychophysical studies of vision, various investigators have proposed algorithms to model limited aspects of perception. Some of the psychophysical studies investigated the binocular nature of vision. Bela Julesz devised random dot stereograms which created an illusion of depth perception [26]. Such investigations show that depth perception does not need to depend on information such as the identity of more complex objects, but rather can proceed solely on pattern matching alone.

Roger Shepard and Jacqueline Metzler investigated the psychophysics of pattern recognition [27]. In their experiment, they presented subjects with groups of line drawings of simple block objects. The presented objects differed by rotation and/or mirror inversion. They found that recognition time varied directly with the amount of rotation difference between the images.

These psychophysical studies provide information about isolated modules that the human visual apparatus uses to identify information contained within scenes. They direct the search for algorithms toward ones that react in the same manner as observed experimentally.

A second approach taken was that of neurophysiological studies of retinal cells [28]. Barlow studied the ganglion cells of the frog retina. Such ganglion cells were monitored as their receptive field was stimulated, noting which responses activated the cell. Such cells, when stimulated by an appropriate visual stimulus, often invoked a feeding response in the frog; the frog would jump toward the stimulus and snap its mouth. This response is important, for it shows that, at least in the frog, a large degree of information processing has occurred at the level of the retina, and at the level of a single neuron. These observations lead Barlow to state:

> *The cumulative effect of all the changes I have tried to outline above has been to make us realize that each single neuron can perform a much more complex and subtle task than had previously been thought. Neurons do not loosely and unreliably remap the luminous intensities of the visual image onto our sensorium, but instead they detect pattern elements, discriminate the depth of objects, ignore irrelevant causes of variation and are arranged in an intriguing hierarchy.*

This lead to perhaps an overstatement of Barlow,

> *A description of that activity of a single nerve cell which is transmitted to and influences other nerve cells and of a nerve cell's response to such influences from other cells, is a complete enough description for functional understanding of the nervous system. There is nothing else "looking at" or controlling this activity, which must therefore provide a basis for understanding how the brain controls behaviour.*

The ability to monitor the signal of individual neurons lead investigators to trace the behavior of neurons at successivly deeper neural levels. Hubel and Wiesel studied the nature of neurons receptive fields at the level of the visual cortex of the cat, finding organization into functional columns [35].

These statements directed neurophysiological studies for the following decades. Investigators searched for cells containing 'units' of information. Gross, Rocha-Miranda, and Bender found cells in the inferotemporal cortex which were activated when a hand was present in the visual field [29]. They were a stepping stone on the way to finding the elusive grandmother cell, a cell that is activated when one's *grandmother comes* into view.

The psychophysical and neurophysical studies were useful in describing the observed behavior at a cell or an individual level, but were only that. They did not lead to an understanding or and explanation of the process of vision.

1.3 Artifical visual systems

Artifical visual systems have simplified the problem by dividing the visual process into low level and high level vision. The term low level vision is used to describe the extraction of basic information from the visual scene. It is the ability to characterize luminosity, depth, color, texture, and depth perception. It makes generic assumptions about the information being processed, and is iconic in nature. One specific example is in extracting shape information based on shading of objects within the scene. This information provides an intermediate representation of the visual scene suitable for high level visual processes to work.

One distinguishing feature of high level vision is that it emphasizes contextual knowledge, i.i. matching to exprected models, as opposed to low-level vision, whose processing is iconic, uniform and simple.

Artifical visual systems have been created to implement selected aspects of the visual process. Several different approaches have been taken to achieve this goal. The first is to try to improve the low-level visual information being extracted from the raw image. This approach, for example, tries to improve the performance of edge-detector operators that are applied to the raw image. This approach leads to the generation of sets of lines or other image information which are used to describe the image.

The next step or goal of the process is in how to use this information to generate information about the actual content of the scene. If the initial scene is limited in scope, such as a collection of toy blocks, then there are defined algorithms to generate one from the other.

An approach by Waltz achieves this by cataloging all possible verticies contained within the image [32]. By examining the type of vertex, and by grouping sets of vertices together according to the edges with interconnect them, it is possible to fit an object or limited set of objects to the edge information.

Several groups have extended this type of "catalog" approach to images with contain non-convex polyhedra [31]. Waltz has also extended this approach by the addition of shadows [32]. These programs are able to examine groups of possible vertices, eliminate those which are spurious (due to shadows, object overlap) and label the remaining vertices, objects, and planes. They do so at the expense of a combinatorically expanding catalog of possible edge and vertex tables. The idea of using a stored model of a shape to assist the recognition process was introduced by Roberts [30], in a computer program that produced edge descriptions of images built with cubes and wedges.

Such approaches rapidly break down, however, for more complex, realistic imagery. The approach is also inadequate when image information is incomplete, or possibly inconsistent.

1.4 Computational Theory

David Marr has formalized the vision problem by casting it as an of information processing task [18]. In doing so, he separated the problem into three distinct levels which, while independent, interact. The separate levels are a) computational theory, b) representation and algorithm, and c) hardware implementation.

The first, computational theory, concerns the abstract goals of the process to be achieved. It formally states the desired transformation to be achieved. With regard to stereopsis, the goal is to extract the notion of depth from the scene. The details of how the extraction is to occur is not specified at this level.

Th' second level concerns representations and algorithms. The same information may be represented in many different forms. This representation may be tailored to suit the desired transformation, and may vary at different steps in the processing task. Some examples of the representation of visual information are gray-level intensity images and semantic networks of related objects within the scene.

The second level also concerns the algorithms used to implement the desired transformation specified in the first level, and how these algorithms interact to share information. The algorithms and the representation of the data are intimately coupled. The algorithm for longhand multiplication is straightforward when the numbers are represented in a base ten system, but quite difficult when the data is represented in Roman numerals. The algorithms in a visual system are the programs which attempt to extract information, such as edges, from the visual data.

The third level concerns the details of the hardware upon which the algorithms are to be implemented. The hardware may range from electrical transistors, connected in serial or parallel configurations, to the biological neurons.

The levels are loosely linked. The set of algorithms available are, to an extent, constrained by the nature of the hardware available. Similarly, there is ambiguity as to which level a task must be assigned. At what level should neuropsychological phenoma be assigned, at the detector (hardware) level, or at an algorithm level, or does it arise from an interaction between levels? Additionally, the level at which various tasks are processed may shift with the changing nature of goals, algorithms, and hardware.

Within this multi-level context, it is possible to roughly separate the contributions of the various fields. Psychophysics may be loosely linked to the level of information representation and the algorithms applied to it. Different visual algorithms fail in markedly different manners when pushed to extremes. These failures observed in psychophysics may help guide the search at the algorithm level. Neuroanatomy more closely applies to the hardware that implements the algorithms and stores the representations.

This approach allows one to analyze why various visual systems fail. Systems may fail because their goal was limited (level 1), the algorithms or data representations are not flexible enough to allow the goal (level 2), or the hardware is not appropriate to the task (level 3).

2 Recognition

One of the processes that a visual system must achieve is the recognition of objects observed in the visual field. The recognition may be described as occurring at several levels, starting from information provided by the low level visual processes. Object recognition is a process of constructing an abstract description of the data, so that a small amount of information (e.g. a "car") can effectively summarize a large amount of data (e.g. millions of bits comprising the image of a car) for purposes of high-level reasoning.

One method of accomplishing recognition is to decompose an object into its parts, and solve the problem by recognizing the parts. The matching is relational in that recognition is contextually dependent on the disposition of other parts. In fact, the grouping of image data into parts and the matching of these parts must usually proceed simultaneously and interact with each other. Such recognition processes must function when limited information is available, such as when shadows or occlusion obscure parts of the image.

2.1 Matching

The process of recognition consists of matching the observed part within the image to a stored database of known objects. Such internal descriptions of objects may concern what are thought to be important information about the object, such as size, color, texture, shape, or other available information.

What remains is to match the observed object to the list of internal descriptions available. Problems arise, however, in that several objects may partially fit, creating confusion. One approach often taken is to limit the input objects or the description parameters so as not to create the confusion of overlap.

Another possible approach is in extending the scope of the matching process of an individual object to encompass the matching of other objects within the scene. This is often possible since other objects within the scene contain information which may bear upon the recognition of the original object (a head may help to recognize a hand instead of a paw). In doing so, an additional level of matching is uncovered, that of matching groups of internal descriptions to possible groups of objects within the scene.

A limit is placed on the combinatoric nature of the memory by separating ob,ects into their constituent parts. Multiple objects may be created from a limited number of constituent parts, and recognition then becomes a two level process. The first is in recognizing the limited number of parts, and the second in matching a group of the parts to groups of parts in the image memory. These two levels may interact to help the matching process of the other.

2.1.1 Hierarchy

Models may be organized into hierarchies in order that large numbers of objects may be stored in the database. Two types that are useful are compositional hierarchies and specialization hierarchies. Compositional ("part-of") hierarchies impose specific top-down restrictions on the combinations of low-level elements which need to be considered, thus reducing search cost. Specialization ("type-of") hierarchies reduce work by allowing incremental recognition. To do recognition, the vision system must dynamically find groups of parts in the image whose internal relations satisfy the definitions imposed by the stored models in the parts hierarchy.

2.1.2 Invariance

A goal of a recognition system applied to a nontrivial domain is that it must recognize patterns and objects regardless of simple transformations such as translation, rotation, scaling, and perspective. The recognition should be able to operate with partial information of the object, and should be able to recognize a large number of objects. It is desirable to avoid a visual memory which achieves this latitude by separately storing every conceivable view of every object to be recognized.

2.1.3 Invariant parameter matching

Matching may be approached through parameter matching. Invariant aspects of the objects or groups of objects may be used to construct a list of parameters describing each object or collection of objects. Matching then becomes a table lookup problem to recall the label associated with the set of parameters. Such approaches must decide what are the invariant aspects of objects which are used to generate the parameters.

2.1.4 Parameter optimization matching

The internal images may be stored not as objects but as mathematical functions describing the image, with a set of tunable parameters. Matching may then proceed by fitting the parameters to the observed objects. This can be used when there is a known object to be found, though it may vary from the internal object due to size or rotation. These techniques work well when the input data objects belong to a limited set of known objects, and the degrees of variation are limited. This technique has been employed by Marr and Nishihara [33] in matching articulated collections of cylinders to match observed data.

2.1.5 Graph theory

Matching problems may also be formulated in terms of graph matching. The internal description of a group of objects may be viewed as a relational graph structure containing the objects as vertices which are connected with arcs, describing some relationship between the parts. The arc may be used to describe a physical connection, such as the object arm is connected to the object body. Such objects are termed graphs.

If the object contained within the scene is also described in terms of a graph, then the recognition process at the top level consists of matching the input graph to a set of stored graphs. The matching used is broken down in the graph matching field into that of a) graph isomorphism: matching graph A to graph B with a one-to-one relationship, b) subgraph isomorphism: matching graph A to a subgraph of B, and c) double subgraph isomorphisms: matching subgraphs of A to subgraphs of B. Again, the problems are complicated by the possibility of missing or extra parts, and the addition of the notion of 'goodness of fit' parameters which may regulate the matching of nodes of the graphs.

One aspect of graph matching which is not immediately apparent is that of the required computational complexity of the algorithms. Most graph matching algorithms are known as NP-complete, all known solutions require a computational power that is, at worse case, exponential in relationship to the size of the graphs needed to be matched. This exponential nature of recognition is not observed in visual psychophysical studies. Indeed, scenes with detailed information are often recognized faster than scenes with less detailed information.

3 Artificial Neural Networks

Artificial neural networks are collections of computational units, termed "neurons," which are interconnected. The computational units perform a specified transformation of their input(s) to their output. Input values may also have "weighting factors" associated with them. The derived output then becomes the input for some specified set of other neurons. Styles of networks vary according to the actual transformation performed by the neurons (binary, linear, sigmoidal) and the nature of the interconnections (layered, symmetrical,complete). One model we have used is the Hopfield-style analog neural network [2]. This model uses a sigmoidal transformation function and has symmetrical interconnections. Starting values and interconnection weights are set, then the neural network is allowed to "relax," seeking a stable state. The output is the set of final resting output values of the neurons.

We have chosen to implement the recognition algorithms in the form of optimization of objective functions implemented in neural networks. Objective functions, which specify how graph matching is to occur, are attractive ways to formulate visual algorithms. Such formulations are concise, being expressible in several lines of algebra. Notions of hierarchy are easily incorporated in objective functions.

An additional advantage of objective functions is that they may be emulated in function by a Hopfield style analog neural network [2]. Solving the objective functions becomes a problem in optimization theory: finding a solution consists of finding minima of the objective function. Analog neural networks may be designed to achieve this objective, with the additional advantages of error tolerance of both the hardware and of the data, and the advantage of learning abilities of neural networks.

3.1 Learning

Learning may be viewed as the process of modifying a pre-existing response so that it better fit a desired response to a stimulus. Learning may occur when the stimulus and the desired response are known. It then becomes a matter of adjusting, via a learning algorithm, the internal parameters which control the response. Generalization of the learned response occurs when the modified algorithm is applied to new input data.

Separate from the process of recognition is the process of learning. Learning in vision systems may consist of adding new objects to the stored image database, or modifying the parameters of already stored images. Learning may also consist of extracting abstractions from groups of items. The nature of neural networks allows learning to occur, although the process of learning a database is more complex than in applying that information. Specifically, we concentrate on learning to discriminate match metrics that characterize individual models, fine-tuning the image recognition process.

Experiments by Hurlbert and Poggio [21] have shown that neural networks are capable of learning a color algorithm. Using optimal linear estimation to synthesize the algorithm and a large paired set of input and desired output images, they were able to train a neural network to perform a "lightness algorithm." Interestingly, the learned algorithm resembled a "lightness algorithm" previously proposed by Land [34]. One limitation of the learning algorithm employed is that the derived algorithms are strictly linear in nature, which may not be a valid assumption of the desired algorithm.

Another approach to learning in neural networks is the back-propagation learning algorithm. The neural network model consists of an input layer, an output layer, and possibly intervening layers. Training vectors consist of an input pattern and a desired output pattern. For each training vector, an error signal is generated between the desired output and the actual output, and this signal is propagated from output to input to reduce the error signal. One advantage of this algorithm is that it operates on neurons with nonlinear output functions, thereby allowing the learning of non-linear algorithms.

The back-propagation technique has been applied to the task of deriving a shape from shading visual algorithm. Curvature is a basic property of objects within a visual scene. Shading of objects depends on multiple factors, such as illumination, orientation, and curvature. Various analytical processes exist to obtain the curvature of an object in an image from the observed shading. Sejnowski and Lehky [23] have shown that it is possible to train a neural network to extract curvature information from shading information. They used a three-layer neural network and applied the back-propagation technique using a training set of 2000 images of parabolic surfaces. One interesting aspect of this experiment was that the "neurons" of the hidden middle layer of the neural network specialized in their fuctions, with regard to orientation of curvature and the sign of the curvature.

3.2 Combinatorial optimization

Many of the algorithms used in recognition are combinatorial in nature. Computational time to solve the problem increases exponentially in respect to the size of the problem to be solved, and must be overcome in order to construct practical visual recognition systems.

A highly abstracted version of graph matching has been used by Hopfield and Tank to solve the travelling salesman problem (TSP) [3]. Simply stated, the goal is to find the shortest path connecting a set of points, with the constraint that every point be visited only once. Such a problem is easily restated as an objective function, and the problem becomes one of minimizing, or optimizing, the objective function. Tank implemented the objective function with analog style neural networks. Such networks, while not usually finding the best answer, quickly find quite good solutions to the problem.

Shown in Figure 1 are two graphs. We may represent a graph by a sparse binary connection matrix whose ijth element is unity if node i is connected to node j, and is zero otherwise. The two graphs shown below are represented by symmetric connection matrices $G_{\alpha\beta}$ and g_{ij}. A matching matrix $M_{\alpha i}$, where $0 < M_{\alpha i} < 1$, of dynamic variables, which we call "match neurons", represents the correspondence between nodes α and i of the two graphs to be matched. The $M_{\alpha i}$ are elements that make decisions. Their activation determines whether α is matched to i ($M_{\alpha i}$ activated) or not matched ($M_{\alpha i}$ not activated.) $M_{\alpha i}$ neurons are analog, and may assume any value between 0 and 1. These intermediate values, obtained when the neural network has not reached a stable state, are the strength of the hypothesis that α and i are matched.

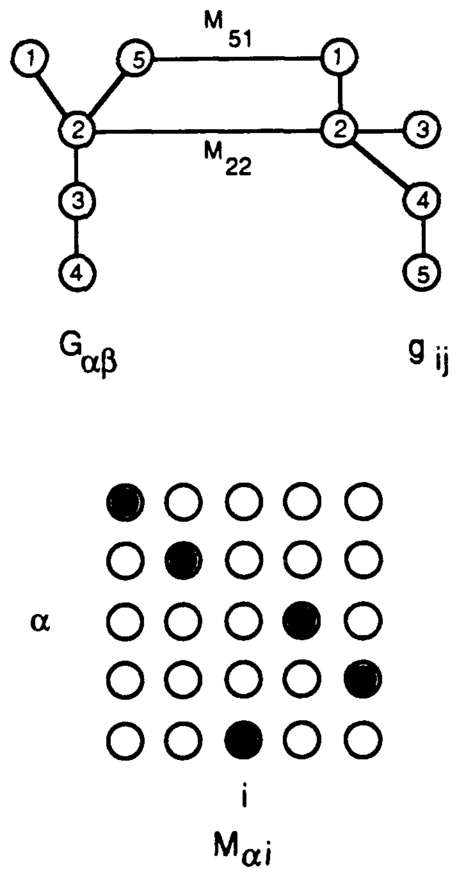


Figure 1: An illustration of a graph matching problem

Parts being matched form local consistency rectangles [7]. Such a rectangle consists of a pair of G-linked model nodes connected by their respective match neurons

M to a pair of g-linked data nodes. An example of such a rectangle is illustrated in Figure 1, consisting of the series G_{52} M_{22} g_{21} M_{51}. A consistency rectangle is a structure that supports matching based on contextual evidence. For example, the match M_{51} between node 5 in G and node 1 in g is enhanced because both nodes have similar neighborhoods (nodes 2 in G and 2 in g) and the neighbors themselves match ($M_{22} = 1$). A simple objective (or energy) function maximizes the number of consistency rectangles by rewarding the activation of such collections:

$$E(M) = -\sum_{\alpha\beta}\sum_{ij} G_{\alpha\beta} g_{ij} M_{\alpha i} M_{\beta j}.$$

Versions of this objective function for graph matching have been proposed by Hopfield [15] and von der Malsburg [16]. This may be understood as follows: given a connection between α and β and a connection between i and j, the global energy term may be minimized by strengthening $M_{\alpha\beta}$ and M_{ij}. The M neurons represent a possible correspondence between pairs $\alpha\beta$ and pairs ij. Without any limiting constraints, such a term merely maximizes the total number of consistent rectangles.

Other terms are necessary to reflect the constraint of one-to-one matches between model nodes and data nodes, limiting the proliferation of consistency rectangles:

$$\sum_{\alpha}(\sum_{i} M_{\alpha i} - 1) = 0$$

$$\sum_{i}(\sum_{\alpha} M_{\alpha i} - 1) = 0$$

These may be understood as follows: the sum of M neurons connected to any particular α (or i) must equal unity. If the M were to assume binary values, then these rules require that each α (or i) be matched to one and only one i (or α). Such constraints may be expressed as penalty terms in an objective function and summed:

$$E(M) = c_1 \sum_\alpha (\sum_i M_{\alpha i} - 1)^2 + c_2 \sum_i (\sum_\alpha M_{\alpha i} - 1)^2$$

Another term is used to encourage the match neurons to assume binary values:

$$E(M) = \sum_{\alpha i} M_{\alpha i}(1 - M_{\alpha i}).$$

This term is minimized as each $M_{\alpha i}$ approaches 0 or 1. Another term that is used in Stickville is the analog gain term [2]:

$$\begin{aligned} E(M) &= \sum_{\alpha i} \int^{M_{\alpha i}} g^{-1}(x)dx \\ g(x) &= \frac{1}{2}(1 + tanh(x)) \end{aligned}$$

where g is a sigmoidal gain function. Such a term acts as a barrier to prevent the allowed range of M from being exceeded by increasing the energy needed to approach the edges of the range. In a discrete simulation of an analog neural network, this has the effect of reducing the step size as the edge of the boundary is approached.

In Figure 1, the match matrix is depicted as a 2-D array of match neurons (circles) whose values are proportional to the radius of the shaded region of circle. The solution depicted, a 1-0 permutation matrix, satisfies the various constraints and is consistent with an optimal matching of the two graphs shown in the figure.

As shown in [2], such objective functions specify the connection weights and biases for a network of neurons $M_{\alpha i}$. Suitable dynamics, discussed in later sections, may be specified so that the network evolves to find a local minimum of the objective function.

The dynamics approximate a gradient descent on the energy surface created by the constraint functions. The gradient descent is distorted by the barrier terms near the boundaries. A discussion of the effects of the barrier term on the network dynamics may be found in [9].

3.3 Varieties of Neural Networks

The details of implementation of neural networks differ with respect to the behavior of the individual neuron (binary, analog, linear, nonlinear), the nature of connections between neurons (complete interconnections, layered approach, directed), and the selection of learning algorithms used. We have chosen to use Hopfield-style analog networks with the neurons using sigmoidal output functions.

4 A Simple Domain: Stickville

We attempt recognition in Stickville, a simple domain of connected assemblages of directed linear "sticks", each possessing a base and tip [4], as shown in Figure 2. Ambiguity exists only for the initial root data stick, the first point entered being defined as the base (circle at end of stick in Fig 2). All other sticks have their base located on their parent stick. In Stickville, we abstract objects by main parts, i.e. designated sticks defined in the model base. Thus a *plane* is abstracted by a single main part *fuselage*. The parameters of *fuselage* become the parameters of *plane*.

Data is represented by a sparse connection matrix ina_{ij} whose value is unity if stick i is connected to stick j. The matrix ina is a symmetric matrix, with $ina_{ij} = ina_{ji}$. This is done to allow any part to assume the role of a local root stick, thereby becoming the main part. Models may then match to a connected subset of a connected assemblage of sticks in the data base. In such a case, a *jet* may be recognized even though it is attached to a *runway*, which is not in the model base. A parent data stick may be connected to many offspring data sticks, but each data stick must have only one parent data stick. This requirement disallows the sharing of a common data stick between two parent sticks. For example, a *wing* may not be attached to two *planes*. Thus each connected assemblage of sticks in the data forms a tree structure.

Appended to each connected stick pair is a set of three parameters $\vec{P_{ij}} = (r_{ij}, \theta_{ij}, q_{ij})$ that measure, respectively, relative size, angle, and location of the attachment point, Figure 2(c). Relative length r_{ij} is defined as $log(\frac{length_j}{length_i})$, with the range $[-\infty, +\infty]$. The relative angle θ_{ij} is defined as the positive acute angle separating i, j, the allowable range being $[0, \pi/2]$. By this definition, $\theta_{ij} = \theta_{ji}$. There is some ambiguity in our definition of θ, since stick j may assume 4 orientations with respect to stick i, all having the same θ_{ij}. The relative attachment point q_{ij} is defined as the fractional distance in terms of i from the base of i that the base of j is attached. The allowable range is $[0, 1]$. For the case where stick i abuts stick j, then we set $q_{ij} = 0$. The $\vec{P_{ij}}$ are presumed precomputed for the input data set.

Models are specified in a similar manner (Figure 2(b)). A sparse binary matrix $INA_{\alpha\beta}$ is unity if model stick β is "part of" model stick α. For Stickville, the INA matrix encodes a tree structure (no shared parts). The matrix $INA_{\alpha\beta}$ is not symmetric, unlike ina_{ij}. This is because the root part is presumed known, as is the order of hierarchy from parent to offspring sticks. Implicit here is the abstraction of a collection of parts to a single main part. Thus several stick models constituting an "airplane" may have a single stick, "fuselage", as a main part. This is shown in Figure 2.

Appended to each INAed model pair is a parametrized function $F_{\alpha\beta}(\vec{P_{ij}})$ that

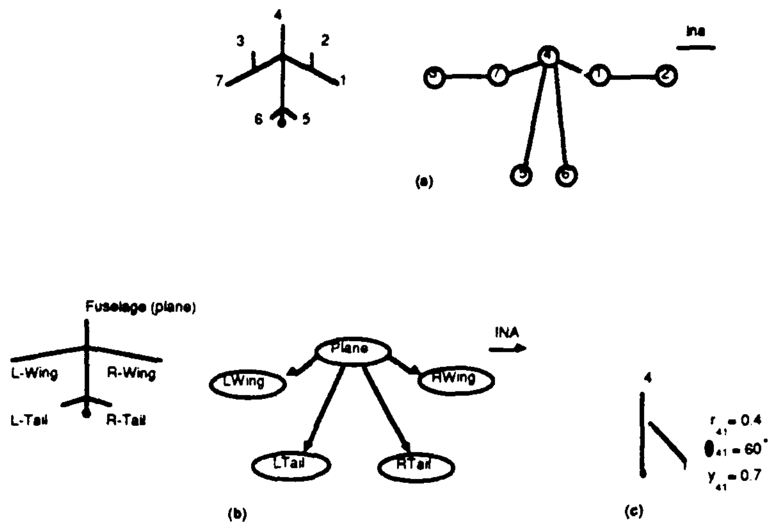


Figure 2: Definitions in Stickville.

specifies how well the parameters of a stick pair ij fit requirements for models α and β. For Stickville, F was a simple quadratic that assumed a positive value of unity for good matches and negative values for bad matches. For example, for relative size,

$$F_{fuselage,wing}(\vec{P}_{4,1}) = (1 - ((r_{4,1} - r^*_{fuselage,wing})/\sigma_{fuselage,wing})^2)/(\sum_{\beta} INA_{fuselage,\beta})$$

where $r^*_{fuselage,wing}$ is the nominal best value (entered by hand) and $\sigma_{fuselage,wing}$ is the allowable variation for the relative size. For the simulations reported, σ was a constant independent of α and β, with $(\sigma_r, \sigma_\theta, \sigma_q) = (1, \pi/2, 1)$, respectively. The different σ scale their respective F values, allowing them to be summed and expressed as a single value. The *INA* term in the denominator normalizes $F_{\alpha\beta}$ with respect to the number of model parts, allowing models with differing numbers of parts to compete equally.

The functions F are hand designed, but one might suspect that F or possibly its parameters could be learned from training examples [9]. Learning the database itself would be a much more difficult problem [10].

5 From Graph Matching to Model Matching

The objective function for maximizing consistency rectangles may be modified simply to incorporate the "quality of fit" as measured by $F(\vec{P})$:

$$E(M) = -\sum_{\alpha\beta}\sum_{ij} INA_{\alpha\beta} ina_{ij} M_{\alpha i} M_{\beta j} F_{\alpha\beta}(\vec{P}_{ij}).$$

We refer to this as the "rectangle rule." Figure 3 represents such a consistency rectangle in Stickville, with the match neurons depicted as circles. If a stick pair ij fits well with model pair $\alpha\beta$, thus having a positive $F_{\alpha\beta}(\vec{P}_{ij})$, the energy term favors this $\alpha\beta ij$ rectangle, strengthening both $M_{\alpha i}$ and $M_{\beta j}$. For bad fits that have a negative $F_{\alpha\beta}(\vec{P}_{ij})$, the proposed rectangle raises energy and is discouraged, weakening both $M_{\alpha i}$ and $M_{\beta j}$. As in the case for exact graph matching, the rectangle rule must be supplemented by additional "syntactical" terms.

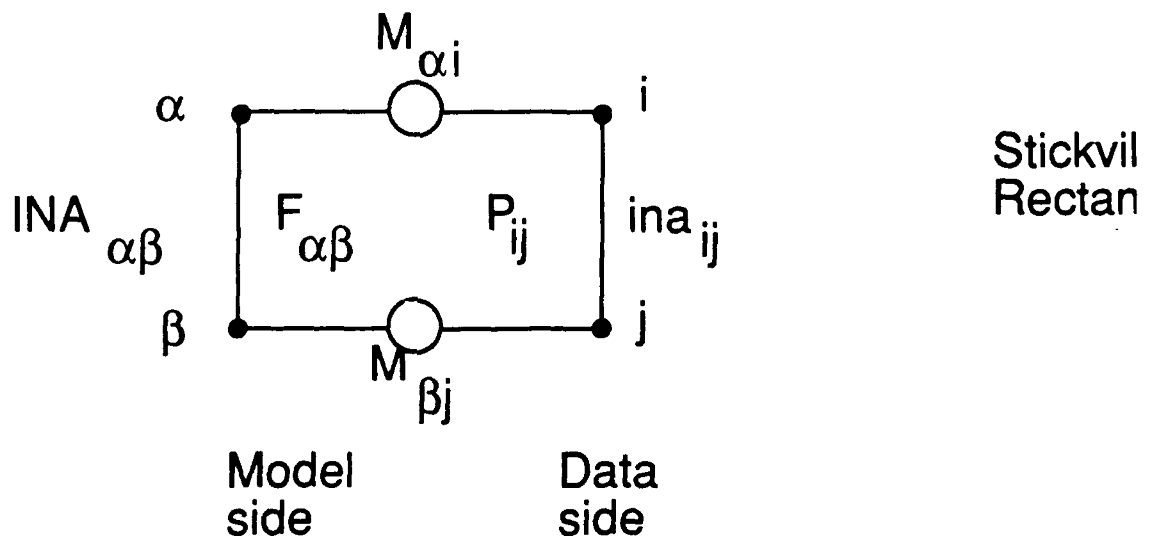


Figure 3: Stickville Rectangle.

Since the syntactic objective functions used in Stickville are somewhat baroque, it is instructive to first consider a series of related simpler problems and express appropriate objective functions for these. A single-level Stickville problem, i.e. one whose data term is linear in M rather than quadratic as in the rectangle rule, consists of the optimal selection of a permutation matrix, directed by the fit constraints which select the best match between graph parts. This is known as a linear weighted match problem, and appears in combinatorial optimization as the Task Assignment Problem. For this problem, we assume an equal number of parts and models and hence a square permutation matrix. In Stickville, the permutation requirement enforces unique matches from each model stick to each data stick. The driving energy term in such a problem is to pick the best combination of such selections, such as the minimum total, while still satisfying the permutation matrix. In Stickville, the minimization seeks the closest fit between parts. An example of a minimized permutation matrix is shown in Figure 4, where the entries are numerical fits.

10	9	▨ 3	6
6	▨ 2	5	6
▨ 1	15	2	10
20	30	8	▨ 2

Figure 4: Minimized Permutation Matrix.

A single level Stickville network is shown in Figure 5. In the diagram at left, the INA links are shown as lines connecting model node α to various child nodes β; the ina links are shown as lines connecting parent stick i to offspring sticks j. Possible matches are shown as lines connecting model and data nodes, with match neurons (circles) sitting on the lines.

To retain Stickville context, we show a possible high-level match $M_{\alpha i}$, but if this is fixed at unity, then the problem reduces to the linear weighted match problem of optimally assigning all child parts β of parent α to all child sticks j of parent stick i. That is, if $M_{\alpha i}$, $INA_{\alpha\beta}$, and ina_{ij} in the rectangle rule are fixed at unity, and fits $F_{\alpha\beta}(\vec{P}_{ij})$ now depend only on β and j (e.g. becomes $F_{\beta j}$), then the rectangle rule becomes an objective function appropriate for the Task Assignment Problem:

$$E = -\sum_{\beta j} M_{\beta j} F_{\beta j}$$

The dynamical variables are $M_{\beta j}$, depicted as a match matrix. Three candidate matches are shown. The left figure depicts all nine possible matches as the lower parts of rectangles. Stickville requires a one-to-one matching between model and

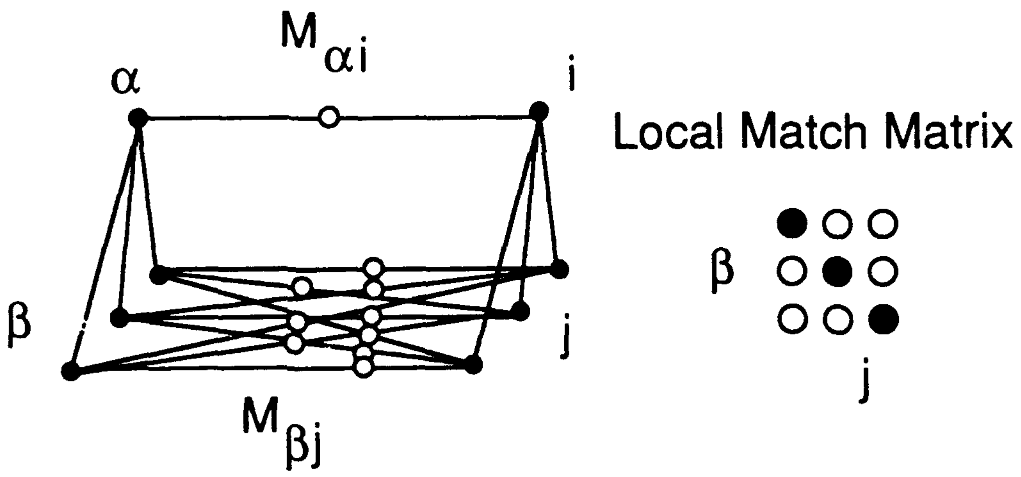


Figure 5: A single level Stickville matrix.

data parts, namely, that each model stick match one data stick, and that each data stick match one model stick. In addition, each neuron should approach a binary value at the solution. The row and column constraints needed to enforce these rules may be expressed as:

$$\sum_{j} M_{\beta j} - 1 \;=\; 0$$

$$\begin{aligned}\sum_\beta M_{\beta j} - 1 &= 0 \\ (M_{\beta j} - 1)M_{\beta j} &= 0\end{aligned}$$

In this case the binary selection rule is not strictly necessary since the problem is one of linear programming. In such a case, the best answer to the solution is guaranteed to be at one of the boundaries, where $M_{\beta j}$ are 0 or 1, and a linear programming algorithm finds it. (Indeed, the use of the binary constraint may be counterproductive since it encourages the search of the solution space only near the boundaries.)

A more difficult case involves trying to match unequal numbers of model and data sticks, illustrated in Figure 6, where $M_{\alpha i}$ and the *ina* and *INA* matrices are again held at unity. The linear version of the rectangle rule is still the same, but the syntactical constraints now differ. The permutation matrix is no longer square, and either some model or some data sticks will not match. In such cases, a constraint is needed to allow the columns or rows to sum to either 0 or 1. For the row and column constraints above, the equations become:

$$\begin{aligned}(\sum_\beta M_{\beta j} - 1)\sum_\beta M_{\beta j} &= 0 \\ (\sum_j M_{\beta j} - 1)\sum_j M_{\beta j} &= 0\end{aligned}$$

Figure 6 shows the appropriate rectangular match matrix and a possible solution. The left figure depicts all twelve possible matches as the lower parts of rectangle structures. Experiments show that when the above constraints are expressed as third-order penalty terms, globally correct matches are almost always obtained for small (5 x 14) matrices.

We may now escalate the problem by reintroducing dynamic matches $M_{\alpha i}$ of

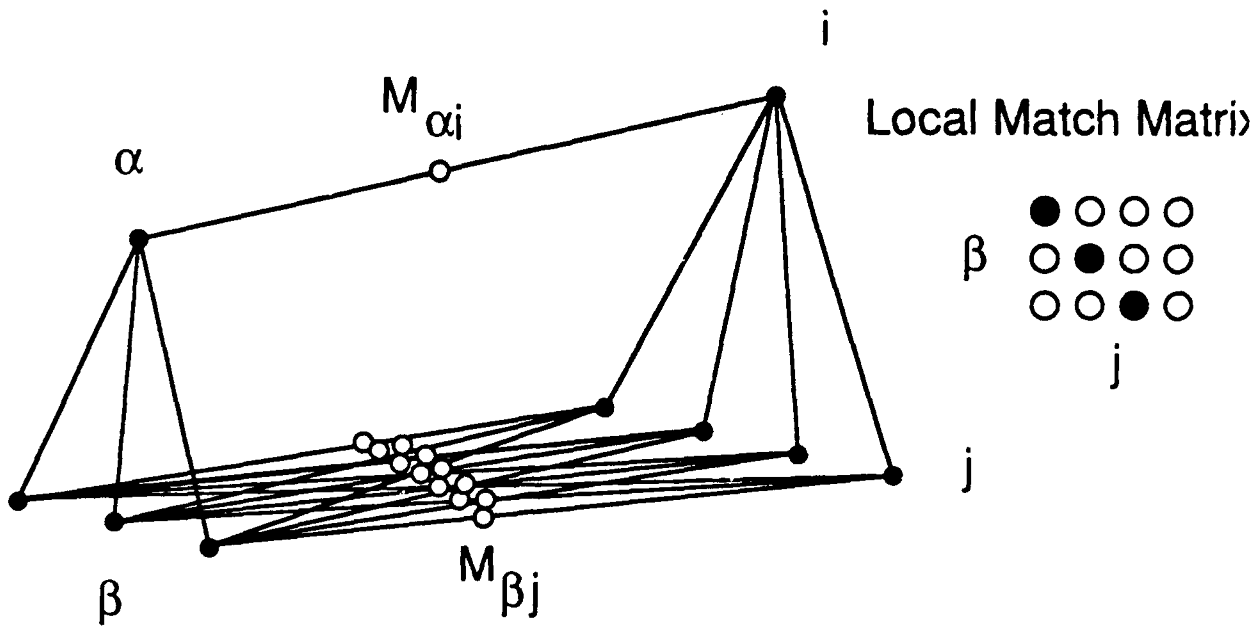


Figure 6: An illustration of a local match matrix βj

parent sticks and models, and hence reinstating the rectangle rule. First consider the case, depicted by Fig 5 but now with dynamic $M_{\alpha i}$, with one parent and equal numbers of sibling parts and models. If we demand reasonably that a sibling match confidence equal that of the parent match, then the row/column constraints previously listed for Fig 5 become:

$$\begin{aligned}\sum_{\beta} INA_{\alpha\beta}M_{\beta j} - ina_{ij}M_{\alpha i} &= 0\\ \sum_{j} ina_{ij}M_{\beta j} - INA_{\alpha\beta}M_{\alpha i} &= 0\end{aligned}$$

The first of these says that several model parts competing for one stick must have activations equal to the parent match. The second of these constraints states a similar constraint for several child sticks competing for the same model. The *ina*, *INA* terms act as "filters" to ensure that this competition takes place among the correct set of competing neurons. (One could imagine a much larger set of neurons than that shown in Fig 5, for example, with the ones depicted being a locally connected, by *ina* and *INA*, subnetwork.) Of course, there may be competing parent matches, (many α and i instead of one). The proper generalization of the constraints above becomes:

$$\begin{aligned}\sum_{\beta} INA_{\alpha\beta}M_{\beta j} - \sum_{i} ina_{ij}M_{\alpha i} &= 0\\ \sum_{j} ina_{ij}M_{\beta j} - \sum_{\alpha} INA_{\alpha\beta}M_{\alpha i} &= 0\end{aligned}$$

Full Stickville incorporates both object-part relations and matches between unequal numbers of model and data sticks. The row and column rules above are thus modified to ones that are actually used in the following Stickville simulations:

$$
\begin{aligned}
(\sum_\beta INA_{\alpha\beta}M_{\beta j} - \sum_i ina_{ij}M_{\alpha i})(\sum_\beta INA_{\alpha\beta}M_{\beta j}) &= 0 \\
(\sum_j ina_{ij}M_{\beta j} - \sum_\alpha INA_{\alpha\beta}M_{\alpha i})(\sum_j ina_{ij}M_{\beta j}) &= 0
\end{aligned}
$$

The binary rule is necessary to help suppress local minima within the solution space, since the problem is no longer one of linear programming. Strictly, the binary rule should be held as a hard constraint, though in the simulations presented here adequate performance was obtained using it as a penalty term.

The preceding constraints were of second-order, with respect to M. To be expressed as additive penalty terms, they must be expressed as a third-order expression. This is shown for the row constraint actually used in Stickville:

$$E = (\sum_\beta INA_{\alpha\beta}M_{\beta j} - \sum_i ina_{ij}M_{\alpha i})^2(\sum_\beta INA_{\alpha\beta}M_{\beta j})$$

The column constraint is expressed similarly as:

$$E = (\sum_j ina_{ij}M_{\beta j} - \sum_\alpha INA_{\alpha\beta}M_{\alpha i})^2(\sum_j ina_{ij}M_{\beta j})$$

The energy curve of such a third order neuron is shown in Figure 7. Here we plot E vs. $M_{\beta j}$ for a single $M_{\beta j}$ and INA, ina set to unity. The horizontal axis is in fractional units of $M_{\alpha i}$ and the vertical axis is in arbitrary energy units. The neuron has the tendency to seek the minima at either 0 or $M_{\alpha i}$. Since $M_{\alpha i}$ is itself changing dynamically, the minimum at $M_{\alpha i}$ shifts dynamically.

Without a driving force, such a Stickville network will be satisfied with all M at ground state, $M = 0$. A constraint is needed that requires the activation of matches

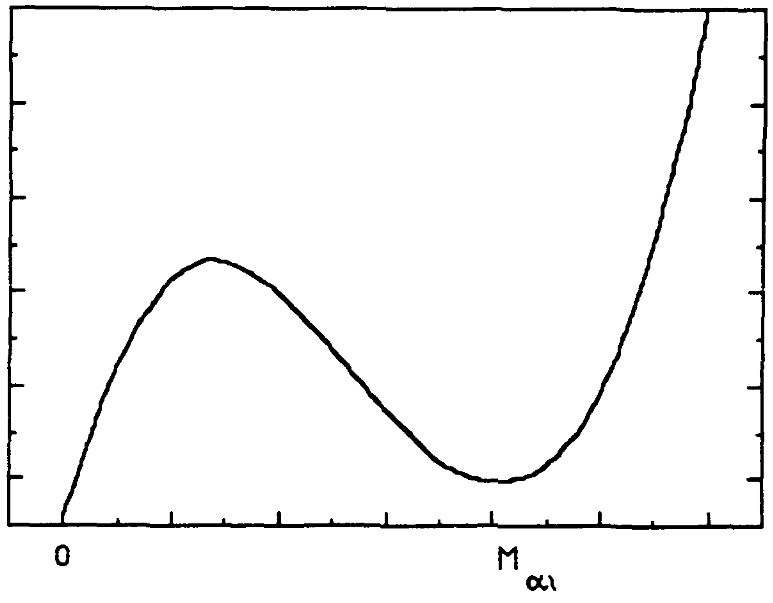


Figure 7: An illustration of a third-order energy curve

consistent with the minimum number of sticks and models. Such a rule may be expressed as:

$$(\sum_{\beta j} INA_{\alpha\beta} ina_{ij} M_{\beta j} - \kappa M_{\alpha i}) = 0$$

where κ is the minimum number of unique matches between model and data sticks. Note that κ may be precomputed and need not be determined dynamically. As a penalty term, we may simply square the above expression.

In the simulations presented here the recognition of Stickville figures is not penalized for extra parts. They simply do not contribute to any of the energy or constraint terms. There is a penalty, however, for missing parts, since the $F_{\alpha\beta}(\vec{P_{ij}})$ objective function is normalized by INA. It is possible to modify the constraint terms to penalize model matches that have additional parts, if it is assumed that an extra part should detract from the quality of match.

6 Introduction of a Specialization Hierarchy

Indexing into a large database of models may be made efficient by the introduction of a specialization (*ISA*) hierarchy. Efficiency is achieved by inheritance of objective functions; work done in matching a parent model need not be repeated in matching a specialization of that model, unless a restriction of parent model parameters is included as part of that specialization. The figure below shows a specialization hierarchy for a set of models in Stickville. Specialization is achieved by narrowing the allowed ranges of parameters between parts, and by the addition of new parts. Thus, in the figure below, a *plane* model becomes a *jet* model by narrowing the wing-fuselage angle so that the wing is swept back. In a ldition, the *jet* model contains engines and elevators not necessary for the *plane* model. For the Stickville simulations presented here, we tested only the case where the addition of new parts was required for specialization. The energy functions here are general, however, and handle both cases.

We define a fixed sparse binary matrix $ISA_{\alpha\beta}$ whose value i nity if model β is a specialization of model α. Our *ISA* matrix forms a tree structure. Models are defined as the entire subset of model sticks that uniquely comprise a figure, such as a *jet*. Thus, in Figure 8, $ISA_{plane,jet} = 1$ and clearly $ISA_{plane,horse} = 0$. The *ISA* matrix is not symmetric.

6.1 An Objective Function for Specialization

The matching matrix and the specialization (*ISA*) matrix may be combined simply: if an assemblage i of sticks matches a model α through the matching matrix, then the matches between the assemblage and each of the model's specializations is enhanced.

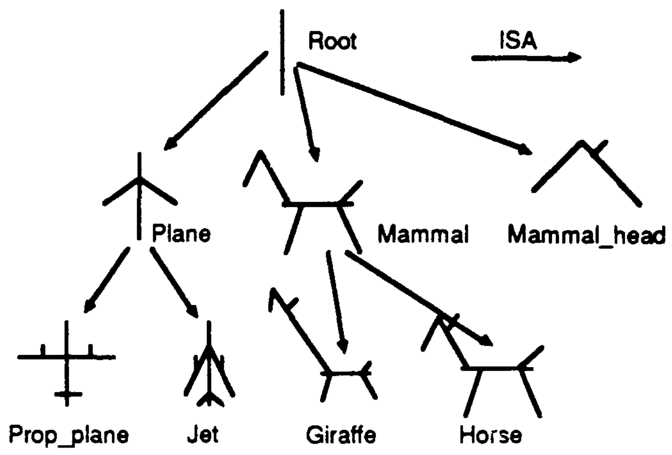


Figure 8: Typical Specialization hierarchy for Stickville

Furthermore, the specializations β compete, on the basis of detailed structure not present in the general model α, to become the unique specialized match to i. The original match between α and i remains. Thus, the intent is to have the specialization hierarchy serve also as a discrimination tree.

Figure 9 illustrates: model α matches stick i. The matches of sibling models β and β' to the same stick are enhanced, but only to the level of the parent match $M_{\alpha i}$. They also compete with one another; the sibling that finds greater support from its parts eventually wins. In steady state, the match $M_{\alpha i}$ and only one of the other two matches to i remains. The structure can also illustrate "bottom-up" indexing. If one of the sibling matches, say $M_{\beta j}$, is on, then the parent match $M_{\alpha i}$ becomes active also, if it had been initially set at a low value. This bottom-up aspect does not correspond to the usual notion of a discrimination tree.

A constraint that achieves all of these effects, while retaining the possibility of no match, is the term

$$(\sum_{\beta} ISA_{\alpha\beta} M_{\beta i} - M_{\alpha i})(\sum_{\beta} ISA_{\alpha\beta} M_{\beta i}) = 0$$

It is not necessary to normalize this function since collections of model sticks compete as models, not individual sticks. If the latter were the case, it would be necessary to normalize the competition rule to allow for equal competition between models with different numbers of parts.

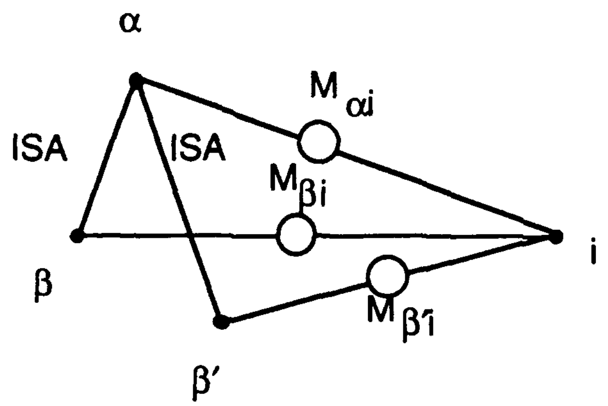


Figure 9: *ISA* specialization structure

The interaction between *INA* and *ISA* is captured in an "*ISA INA*" diamond. An example of the *ISA INA* diamond structure is shown in Figure 10. A consistent rectangle structure is formed between *plane*, *wing* and the data sticks (only two data sticks depicted here), as well as between *prop*, *prop wing* and the same data sticks. If the *plane* and *wing* sticks match, then the specializations *prop* (for "*propellor plane*") and *prop wing* are encouraged by the *ISA* energy term. Verification is ensured by the *prop – prop wing – stick – stick* rectangle.

6.2 Competition between Groups of Sticks

An alternate form of *ISA*-competition was experimented with. Instead of matching individual sticks to model nodes and organizing *ISA*-*INA* as in the "diamond" structure, we may match collections of *ina*-connected sticks directly to collections of *INA*-connected models as shown in the example in Figure 11.

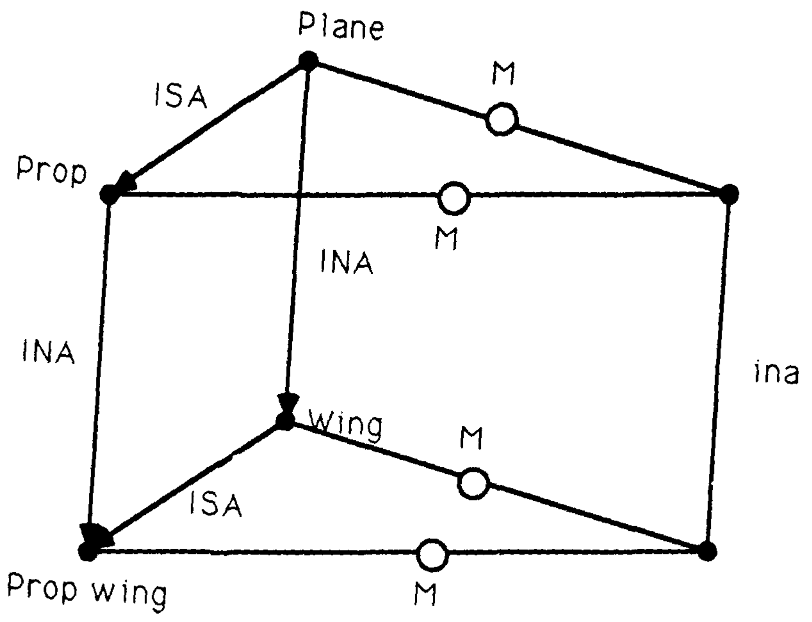


Figure 10: *ISA INA* structure

Models, which we call r-models in this alternate scheme, are indicated in caps, e.g. *PLANE*, and are composed, as shown, of *INA*-connected models of the former sort. r-models are connected by *RISA* links, defined again as sparse binary matrices. An r-model matches an r-stick, which is a labeled collection of connected individual sticks. r-stick 7 is composed of sticks $16-33-44$ in the example shown. The index of r-sticks ranges over all connected assemblages in the data. Matches are represented by neurons $R_{\alpha i}$ (instead of $M_{\alpha i}$) where the indices of R now range over r-models and r-sticks. The diamond organization is gone, but the triangle competition remains:

$$(\sum_{\beta} RISA_{\alpha\beta} R_{\beta i} - R_{\alpha i})(\sum_{\beta} RISA_{\beta i}) = 0$$

Our motivation of *RISA* vs *ISA* comes from results of experiments, described in detail in Section 8.

The new quantities may be defined in terms of the old ones unambiguously. Let i range over all sticks that comprise r-stick i' and β range over all models that comprise

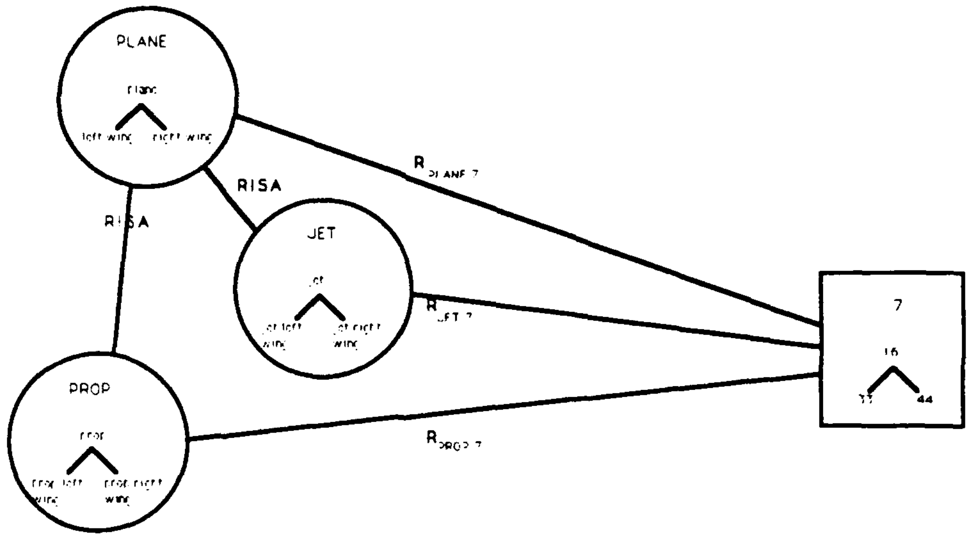


Figure 11: RISA hierarchy

r-model β'. Then,

$$R_{\beta' i'} = \sum_{\beta i} M_{\beta i} \tag{1}$$

Also, *RISA* may be determined from *ISA*. Let α range over models comprising r-models α' and β range over models comprising r-model β'. Then if

$$ISA_{\alpha\beta} = 1, \;\; RISA_{\alpha'\beta'} = 1. \tag{2}$$

For example, if $ISA_{wing,prop\ wing} = 1$ as in the diamond figure, and *wing*, *prop wing* are elements of r-models *PLANE* and *PROP*, respectively, we conclude the $RISA_{PLANE,PROP} = 1$.

7 Unconstrained Optimization with Hopfield Nets

Constraints can be conveniently reformulated into the framework of unconstrained optimization using additive "penalty" terms. This is the approach used by Hopfield in the TSP [2]. For example, the constraint of sibling competition discussed above becomes the additive term,

$$c(\sum_{\beta} ISA_{\alpha\beta}M_{\beta j} - M_{\alpha i})^2(\sum_{\beta} ISA_{\alpha\beta}M_{\beta j})$$

where c is a coefficient that determines the strength of this penalty term. The minima of this term coincides with the desired constraint. A major difficulty with such penalty terms is in the selection of c. A larger c enforces the constraint more rigidly, but at the expense of other penalty terms.

Dynamic variables are identified with neurons, and the connection weights are specified by the objective function. Hopfield [1] has shown that descent to a local minimum of the objective function follows from the equations of motion,

$$\begin{aligned} \frac{du_{\alpha i}}{dt} &= -\frac{\partial E}{\partial M_{\alpha i}} \\ M_{\alpha i} &= g(u_{\alpha i}) \end{aligned}$$

where g is the sigmoidal mapping between M and an internal state u.

8 Constrained Optimization with Hopfield Nets

Unconstrained optimization problems offer the advantages of simplicity and the ability of direct implementation in Hopfield networks. In certain situations, they possess the advantage of seeking compromises between constraints. However, they possess the undesirable properties of not exactly satisfying the original constraint, needing to find suitable weighting factors in relation to the other constraints, and undesirable convergence rates as the constraint strength is increased.

It is possible to solve the constrained problem exactly using the method of Lagrange multipliers [5]. Hard constraints may be expressed in the form $h = 0$. A network implementation is possible [6] where the Lagrange multipliers λ are themselves neurons. The equations of motion are,

$$\begin{aligned} \frac{dM_{\gamma k}}{dt} &= -\frac{\partial E}{\partial M_{\gamma k}} - \sum_{\alpha i} \lambda_{\alpha i} \frac{\partial h_{\alpha i}}{\partial M_{\gamma k}} \qquad (3) \\ \frac{d\lambda_{\alpha i}}{dt} &= h_{\alpha i}(M_{\gamma k}) \qquad (4) \end{aligned}$$

where now we use double subscripting αi to index hard constraints $h_{\alpha i}$ and associated Lagrange neurons $\lambda_{\alpha i}$. The term E includes the objective and remaining penalty terms.

A gradient ascent is performed on the Lagrange neurons, while a steepest gradient descent is performed on M [6]. The method may be modified slightly to ensure positive definiteness and convergence around the minima [6]. The method allows the constraint $h(M)$ to be held exactly at the solution point and also relaxes the selection of constraint strengths.

As stated above, the constrained optimization equations have no provision for the analog gain "barrier" term. A slight modification leads to a constrained form of Hopfield dynamics:

$$\begin{aligned} \frac{du_{\gamma k}}{dt} &= -\frac{\partial E}{\partial M_{\gamma k}} - \sum_{\alpha i} \lambda_{\alpha i} \frac{\partial h_{\alpha i}}{\partial M_{\gamma k}} \\ \frac{d\lambda_{\alpha i}}{dt} &= h_{\alpha i}(M_{\gamma k}) \\ M_{\gamma k} &= g(u_{\gamma k}) \end{aligned}$$

One practical consideration is that each of our penalty constraints may be written as a sum of (at most cubic) monomial terms. The sums determine connection weights between neurons, but no new neurons need be introduced even if there are many terms in the sum. On the other hand, reformulation with Lagrange multipliers requires one Lagrange neuron for each term in the constraint sum, leading to a proliferation of Lagrange neurons. For objective functions with a limited number of constraints, such as the *ISA* fanout rule, the overhead of additional Lagrange neurons and connections is manageable. For objective functions with multiple constraints, such as the binary-value objective function, a Lagrange neuron is needed for each match neuron. Such a requirement doubles both the number of connections and the number of neurons. Depending on the expense of implementing connections and neurons, which itself depends on the neural network implementation method, such a cost may be prohibitive. On the other hand, it could be argued that the network already has $\mathcal{O}(N^2)$ neurons for a database of $\mathcal{O}(N)$ parts or models, so a doubling due to Lagrange neurons is not relatively expensive.

Stickville was implemented using Lagrange neurons to enforce the *ISA* fanout rule. This use of Lagrange neurons led to a modest increase in the number of neurons and connections in our simulations. In this case there is one hard constraint for every occurrence of *ISA* sibling competition as in Figure 9. The hard constraint demands that at the fixed point, one or none *ISA* siblings shall be on (equal to the activation of parent), and all of the others off (equal to 0). We may index the constraint by the indices of the parent neuron αi and thus write the constraint as:

$$h_{\alpha i}(M_{\gamma k}) = h_{\alpha i}(M_{\alpha i}, \mathit{ISA}\ \mathit{siblings\ of}\ M_{\alpha i}) = (\sum_{\beta} \mathit{ISA}_{\alpha\beta} M_{\beta i} - M_{\alpha i})(\sum_{\beta} \mathit{ISA}_{\alpha\beta} M_{\beta i}) = 0 \tag{5}$$

The equations of motion then become:

$$\begin{aligned}
\frac{du_{\gamma k}}{dt} &= -\frac{\partial E}{\partial M_{\gamma k}} - \sum_{\alpha i} \lambda_{\alpha i} \frac{\partial h_{\alpha i}}{\partial M_{\gamma k}} \\
\frac{d\lambda_{\alpha i}}{dt} &= (\sum_{\beta} ISA_{\alpha\beta} M_{\beta i} - M_{\alpha i})(\sum_{\beta} ISA_{\alpha\beta} M_{\beta i}) \\
M_{\gamma k} &= g(u_{\gamma k})
\end{aligned}$$

where,

$$\begin{aligned}
\frac{\partial h_{\alpha i}}{\partial M_{\gamma k}} &= (2\sum_{\beta} ISA_{\alpha\beta} M_{\beta i} - M_{\alpha i})(\sum_{\beta} ISA_{\alpha\beta}) & & if \quad M_{\gamma k} \neq M_{\alpha i} \\
\frac{\partial h_{\alpha i}}{\partial M_{\gamma k}} &= \sum_{\beta} ISA_{\alpha\beta} M_{\beta i} & & if \quad M_{\gamma k} = M_{\alpha i} \\
\frac{\partial h_{\alpha i}}{\partial M_{\gamma k}} &= 0 & & if \quad M_{\gamma k} \; neither \; child, \; parent \; of \; \alpha i
\end{aligned}$$

A similar constraint may be formulated for the *RISA* links. By analogy with Eq. 5, it is,

$$h_{\alpha' i'}(R_{\gamma' k'}) = h_{\alpha' i'}(R_{\alpha' i'}, RISA \; siblings \; of \; R_{\alpha' i'}) = (\sum_{\beta'} RISA_{\alpha'\beta'} R_{\beta' i'} - R_{\alpha' i'})(\sum_{\beta'} RISA_{\alpha'\beta'} R_{\beta' i'}) =$$

By Eqs 1 and 2, the above constraint may be re-expressed in terms of the dynamical variables M. The equations of motion follow analogously from Eqs 3 and 4.

Let primed quantities $\alpha'\beta'$ index r-models and *RISA* links, and let i' index r-sticks. Also let the corresponding unprimed quantities (α, β, i) index elements of models, sticks that comprise the correct r-models, r-sticks. Then we may write $h_{\alpha' i'}$ in terms of match neurons:

$$h_{\alpha' i'}(M) = (\sum_{\substack{\alpha \epsilon \alpha' \\ \beta \epsilon \beta'}} \sum_{i \epsilon i'} ISA_{\alpha\beta} M_{\beta i} - \sum_{\substack{\alpha \epsilon \alpha' \\ i \epsilon i'}} M_{\alpha i})(\sum_{\substack{\alpha \epsilon \alpha' \\ \beta \epsilon \beta'}} ISA_{\alpha\beta} M_{\beta i}) = 0$$

From the form of the constraint, it is seen that entire groups of neurons compete rather than individual *ISA*-siblings.

The equations of motion become:

$$\begin{aligned}
\frac{du_{\gamma k}}{dt} &= -\frac{\partial E}{\partial M_{\gamma k}} - \sum_{\alpha' i'} \lambda_{\alpha' i'} \frac{\partial h_{\alpha' i'}}{\partial M_{\gamma k}} \\
\frac{d\lambda_{\alpha' i'}}{dt} &= \Big(\sum_{\substack{\alpha\epsilon\alpha' \\ \beta\epsilon\beta'}} \sum_{i\epsilon i'} ISA_{\alpha\beta} M_{\beta i} - \sum_{\substack{\alpha\epsilon\alpha' \\ i\epsilon i'}} M_{\alpha i}\Big)\Big(\sum_{\substack{\alpha\epsilon\alpha' \\ \beta\epsilon\beta'}} \sum_{i\epsilon i'} ISA_{\alpha\beta} M_{\beta i}\Big) \\
M_{\gamma k} &= g(u_{\gamma k})
\end{aligned}$$

where,

$$\frac{\partial h_{\alpha' i'}}{\partial M_{\gamma k}} = \Big(2\sum_{\substack{\alpha\epsilon\alpha' \\ \beta\epsilon\beta'}} \sum_{i\epsilon i'} ISA_{\alpha\beta} M_{\beta i} - \sum_{\substack{\alpha\epsilon\alpha' \\ i\epsilon i'}} M_{\alpha i}\Big)\Big(\sum_{\substack{\alpha\epsilon\alpha' \\ \beta\epsilon\beta'}} M_{\alpha i}\Big)\Big(\sum_{\substack{\gamma\epsilon\alpha' \\ \beta\epsilon\beta'}} \sum_{i\epsilon i'} ISA_{\alpha\beta}\Big) \qquad \begin{matrix} \text{if } M_{\gamma k} \neq M_{\alpha i} \\ \alpha\epsilon\alpha', i\epsilon i' \end{matrix}$$

$$\frac{\partial h_{\alpha' i'}}{\partial M_{\gamma k}} = \sum_{\substack{\alpha\epsilon\alpha' \\ \beta\epsilon\beta'}} \sum_{i\epsilon i'} ISA_{\alpha\beta} M_{\beta i} \qquad \begin{matrix} \text{if } M_{\gamma k} = M_{\alpha i} \\ \alpha\epsilon\alpha', i\epsilon i' \end{matrix}$$

$$\frac{\partial h_{\alpha' i'}}{\partial M_{\gamma k}} = 0$$

The above dynamics are complicated, but may be understood more easily with the aid of Figs. 9 and 11. In Fig 9 individual neurons that match the same stick and are also *ISA* siblings compete via the constraint $h_{\alpha i}$ of Eq 5. Strictly speaking, the constraint would be satisfied if the sum of the neurons equaled unity, but other constraints force one (or none) siblings to the value of the parent neuron and the others to zero. In Fig 11, the $R_{\alpha' i'}$ neurons represent the sum of a large cross product, i.e. $R_{\alpha' i'} = \sum_{\alpha\epsilon\alpha'} \sum_{i\epsilon i'} M_{\alpha i}$. The sibling cross products $R_{\beta' i'}$ compete as sums. Now, other constraints force (i) all neurons M in all cross products to zero except for one cross product and (ii) the correct subpermutation within the winning cross product. As seen in the next section, the latter method with r-models often led to more successful results.

9 Implementing Learning

Up until this point the "fit" function $F_{\alpha\beta}$ has been a prespecified function. It is a function which encourages close matches, and discourages mismatches. For Stickville, F took the form,

$$F_{fuselage,wing}(\vec{P}_{4,1}) = (1 - ((r_{4,1} - r^*_{fuselage,wing})/\sigma)^2)/(\sum_\beta INA_{fuselage,\beta})$$

where σ was a constant value. What this equation states is that variation of a part of the data compared to the model is judged by a predetermined notion of good fit. In addition, the same rule is applied without variation to all data parts. This limitation may be removed by employing a family of parameters σ_{ij}, though again the set of parameters is predetermined. An additional limitation of this equation is that it is a unimodular function, generating only one maximia. If we desire that F is capable of matching two or more discrete examples (e.g. an acute angle and an obtuse angle) then F is required to be a multimodal function.

A more desirable solution would be to learn the individual variation rules from a series of training data. The training data would be labelled as a "good" or "bad" fit and allow the neural network to adjust the specific multimodal $F_{\alpha\beta}$.

To implement the learning algorithm I have used a modified back-propagation technique [36]. We wish the untrained network to reject all matches by producing a near-zero output. Trained positive examples should produce a near-one output, sig-

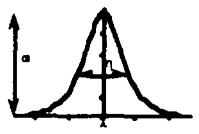


Figure 12: Gaussian output function

nifying a good match. Examples within the local neighborhood of a positive example should produce an intermediate response (0,1) which rapidly tails off towards zero as the difference is increased. Since the desired output function on a local scale can be approximated by a Gaussian function, the output function of the individual neurons composing the learning network were chosen to be a Gaussian function of the form,

$$f_g(x) = we^{-\eta(x-\lambda)^2}$$

This generates a Gaussian function centered on λ, with a width determined by η, and a magnitude determined by w, as shown in figure 12.

Back-propagation techniques are usually applied to networks with monotonic (*e.g.* sigmoidal) output functions only. Moody has shown that similar equations may be applied to networks with hidden-layer Gaussian neurons [37]. These networks retain the gradient-descent nature of the back-propagation networks.

A neural network may be created with an input neuron i, a set of Gaussian neurons in the hidden layer j, and a sigmoidal output neuron o shown in figure 13. The sigmoidal output neuron is used to limit the total output of the network and maps its input to output through the equation:

$$f_s(x) = \frac{1}{2}(1 + \tanh(x))$$

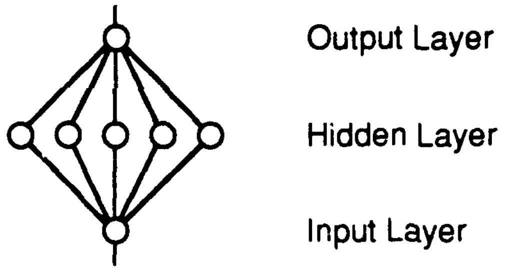


Figure 13: Gaussian neural network

The input to this network is a parameter between an INAed pair of model parts $\alpha\beta$ and a matched inaed pair ij. The desired output is the function $F_{\alpha\beta}$. Training pairs consist of an observed parameter and whether that is allowable (output 1) or not allowable (output 0). A back-propagation technique is used to reduce the error signal generated at the output by adjusting the parameter w. The parameters η, λ are adjusted with reference to the input data to obtain an optimum clustering of the hidden layer neurons.

Within the learning subnetwork each hidden layer neuron j generates a Gaussian output. The aggregate output for a group of hidden layer neurons j takes the form:

$$net = \sum_j w_{jo} e^{-\eta_j (x-\lambda_j)^2}$$

where w_{jo} is the weighting parameter between hidden layer neuron j and the output neuron o. This output then becomes the input net for the single sigmoidal output neuron o.

The learning networks are designed with several hidden layer neurons in order to fine tune the output function F. Parameters w_{jo}, η, λ exist for each of the hidden layer neurons j, where o is the output neuron.

The application of the back propagation equations involves two phases. The first phase consists of presenting an input to the network and propagating it forward to compute the observed output o_o and the generated error signal between the desired output and the observed output. The second phase consists of a backward pass through the network, propagating the error signal through each layer of the network while modifying the neuron parameters to reduce the error signal.

The basic back propagation equation to modify w takes the form:

$$\Delta w_{xy} = l\delta_y o_x$$

This equation states that the weight w between any units x, y should be changed by an amount proportional to the error signal δ_y at the receiving neuron and the output of the sending neuron o_x. l is a parameter that adjusts the rate of learning. The parameter is chosen to allow a resonable rate of learning while avoiding the possibility of oscillation of the network with large values of l. For weights between the hidden layer neurons j and the output neuron o, the equation takes the form:

$$\Delta w_{jo} = l\delta_o o_j$$

The determination of the error signal starts at the output layer. For neurons in the output layer, δ takes the form:

$$\delta_o = (t_o - o_o)f_s'(net_o)$$

where $f_s'(net_o)$ is the derivative of the activation function which maps the total input net_o of a neuron to its output value, in this case the derivative of a sigmoidal mapping function. The term $(t_o - o_o)$ is the difference between the desired output and the generated output for a given input pattern.

In order to limit the response of the hidden Gaussian neurons to achieve local responses, the last equation is modified to the form:

$$\delta_o = (t_o - o_o) f_s'(net_o) f_s(net_o)$$

This equation, graphically displayed in Figure 14 favors local responses while suppressing non-local responses. In the figure, the closer Gaussian response is modified in preference to the remote Gaussian response.

The previous equations adjust the magnitude of the Gaussian neurons to match the test data. The equations to modify the offset and width of the Gaussian involve a feed-forward approach to adjust η, λ of the hidden neurons. For λ, the error term is the difference between the input parameter x and the individual neurons λ's. Changes in the λ_j's are again gated by the response of the individual Gaussian neuron to the input, $f(x)$, or equivalently, $f(net)$. The equations for η are similar in nature. The equations are:

$$\begin{aligned} \Delta\lambda_j &= l\delta_j \\ \delta_j &= (x - \lambda_j) f_g'(x) f_g(x) \\ \Delta\eta_j &= l\delta_j \\ \delta_j &= (x - \eta_j) f_g'(x) f_g(x) \end{aligned}$$

10 Experimental Results

A set of selected experimental results from Stickville appear in Figures 15 through 22. For the following simulations, analog neural networks were digitally simulated, using discrete time steps. The forward Euler method was used to simulate the differential equations.

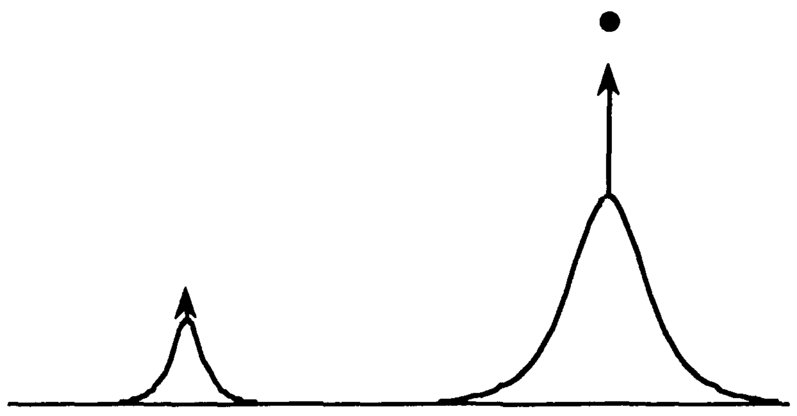


Figure 14: Gaussian function

10.1 Non-Learning Results

In the following examples, the *ISA* fanout rule is enforced as a hard constraint, the Lagrange neurons appearing as "Lagrange *ISA* Constraints." The additional constraint functions, such as binary values and one-to-one matching, are handled as additive penalty terms. While the performance of the network was sensitive to the initial selection of the penalty weights, the weights did not need to be adjusted to solve individual problems. The penalty weights used in the following simulations were:

$$\begin{aligned} row, column\ rules &= 1 \\ binary\ value\ rule &= 1 \\ total\ activation\ rule &= 0.1 \\ rectangle\ rule &= 5 \\ analog\ gain\ term &= 1 \end{aligned}$$

Unless otherwise noted, the network simulations were started with the $M_{\beta j}$ set to small random values on the range [.01, .03].

In the simulations depicted in the figures, there is one Lagrange neuron per *ISA* structure. The leftmost neuron refers to the *ISA* structure higher up in the hierarchy.

We note that the Lagrange neurons are mapped sigmoidally for display (but not computational) purposes.

Figure 15 illustrates a basic incremental match using *ISA* links. In this example, the model consists of a *root*, which specializes to a *plane*, which then specializes to a *prop* and a *jet*. *ISA* neurons are needed for each part at each level of specialization. Specifically, there is a set of 5 Lagrange neurons for the specialization from *root* to *plane*, 1 for each data stick. There are also sets of Lagrange neurons for the specialization of *plane* to *jet* and *prop*, *left wing* to *jet left wing* and *prop left wing*, and *right wing* to *jet right wing* and *prop right wing*. In the figure, the i index of the Lagrange neuron $\lambda_{\alpha i}$ is given by its column location. The α inaex proceeds down the hierarchy: first row is α = *root*, etc. Figure 15(b) shows the network after it has settled on the solution of *prop*. At the solution point, all data sticks match to *root*, some match to *plane* in a specified manner, and all match to *prop* in a specified manner.

Figure 16 points out a weakness of the *ISA* rule. The rule relies only on local information in its decision. For example, the selection between *jet left wing* and *prop left wing*, and the selection between *jet left tail* and *prop left tail* are linked only indirectly by the rectangle rule. Figure 16(b) shows a decision reached that is consistent with the *ISA* rule, but is not a valid terminal node of the model. This inconsistency is avoided if the *ISA* rule is reformulated as the *RISA* rule, where the competition is between the sets of parts forming a model, and not its individual sticks. All the following experiments are performed using the *RISA* rule. The *RISA* neurons are displayed in the same manner described for Figure 15.

Figure 17 illustrates a basic incremental match using the *RISA* rule. The data is a connected assemblage of sticks (an r-stick) that perfectly matches a *jet* model in this case. The individual models label the rows of the match matrix; the appropriate r-models (and their constituents) are *ROOT (root)*, *PLANE (plane, left wing, right wing)*, *JET (jet, jet left wing, jet right wing, jet left engine, jet right engine)*, and *PROP (prop, prop left wing, prop right wing, prop left tail, prop right tail)*. The *RISA* links are $RISA_{ROOT,PLANE}$, $RISA_{PLANE,JET}$, and $RISA_{PLANE,PROP}$. The latter two compete while the former is its own competition. If we label the sole r-stick as, for example, 1, then the only Lagrange neurons are $\lambda_{ROOT,1}$ and $\lambda_{PLANE,1}$. These are displayed at the bottom of the figure.

The figure illustrates the system in terms of models, sticks, and M neurons; the r-versions are implicit but defined as above. At first, all sticks match *root* hence *ROOT*. The $RISA_{ROOT,PLANE}$ link encourages a match to *PLANE*. Several components of r-stick 1, guided by the rectangle rule, find matches to components of r-model *PLANE*. Specifically, sticks 0,1,2 match *plane*, *left wing* and *right wing*. This in turn activates *RISA* competitors *PROP* and *JET*. Through the rectangle rule, they both seek appropriate matches among the sticks of r-stick 1. Eventually, *JET* finds greater support; *PROP* loses out (thus satisfying the *RISA* constraint) and the network reaches a fixed point as shown in Fig 17(c). Among the possible matches to *JET*, the rectangle rule and syntactic constraints select the correct permutation matrix, i.e. the correct matches. In Fig 17(b), intermediate results are shown. Curiously, r-model *PROP* is better matched than *JET* at that point in time, but loses out in the end.

Figure 18 shows a net starting from a high-energy undesirable starting position. In this case, the match neurons from *PLANE RISA* siblings *prop* and *JET* have been set to high value in matching the same stick. This creates an unfavorable energy term from the *RISA* fanout rule. The Lagrange neuron responsible for enforcing this constraint performs a gradient ascent until the rule is enforced by suppressing the matches to *prop*.

Figure 19 shows that the network is capable of performing non-terminal matches. In this case, the data presented to the network is simply a *PLANE* without specializations. The *RISA* links do not proceed past *PLANE* since the fanout rule was formulated as being of low energy when the fanout was $M_{\alpha i}$ or 0. This creates a third-order rule but allows the existence of the two local minima needed to account for the case of a solution as well as no solution.

Figure 20 shows that the network can recognize objects as connected subsets of connected assemblages of sticks. In the case presented, the model to be recognized is a *JET*. The data base clearly contains a *JET* attached to an extraneous stick. The *JET* is recognized, while the match to its parent stick is not.

The next few figures show how *RISA* hierarchies may help speed recognition. As a minimal requirement, one might demand that a model, such as *JET*, may be found more quickly if its *RISA* parent *PLANE* is already matched. This is indeed the case as seen in Figure 21. Here, the network is initialized with correct matches to *ROOT* and *PLANE* already set; the *JET* is found correctly in 20 time steps. If the network is initialized to a random start, however, correct recognition of *JET* takes considerably longer (70 time steps) as expected. Note here that parts of *JET* common to *PLANE*, such as the wings, are rematched as *JET* is recognized. The *RISA* hierarchy is still efficient though since the reverification of *wing* in a *JET* context proceeds more quickly when the (easier) match of *wing* in *PLANE* context is present.

In the next few simulations, a database of *ROOT PLANE JET* is used to test for speed comparisons. Table 1 summarizes results. Here r, p, j stand for *ROOT PLANE JET*, and a capitalized version R, P, J means that the net is initialized with the capitalized model correctly matched. The network starting positions for Rp and Rj are shown in Figure 22. The timing values then give the number of time steps needed for the net to stabilize. Thus, $RPj = 20$ implies that the net is started with *ROOT PLANE* set and it takes 20 time steps to find *JET*. Also, $Rj = 55$ means that only *ROOT JET* are in the model base, and 55 steps are needed to find *JET* if *ROOT* is present. We tested the following cases: RPj, Rpj, Rj, Rp.

RPj	20
Rpj	50
Rj	55
Rp	45

Table 1: Timing Results

Some observations are in order: As expected, $RPj = 20$ is much less than $Rpj = 50$; this is essentially the same result shown in Figure 21. In addition, $Rpj = 50$ is less than $Rj = 55$, so the introduction of an intermediate model speeds recognition, albeit slightly, in this case. Finally it is interesting to compare the time for a fully parallel search, Rpj, with a "serial" search in which a net first finds *PLANE* only, then uses the result as an initial condition to find *JET*. The appropriate comparisons are $Rpj = 50$ vs. $Rp + RPj = 65$, so the parallel one wins here.

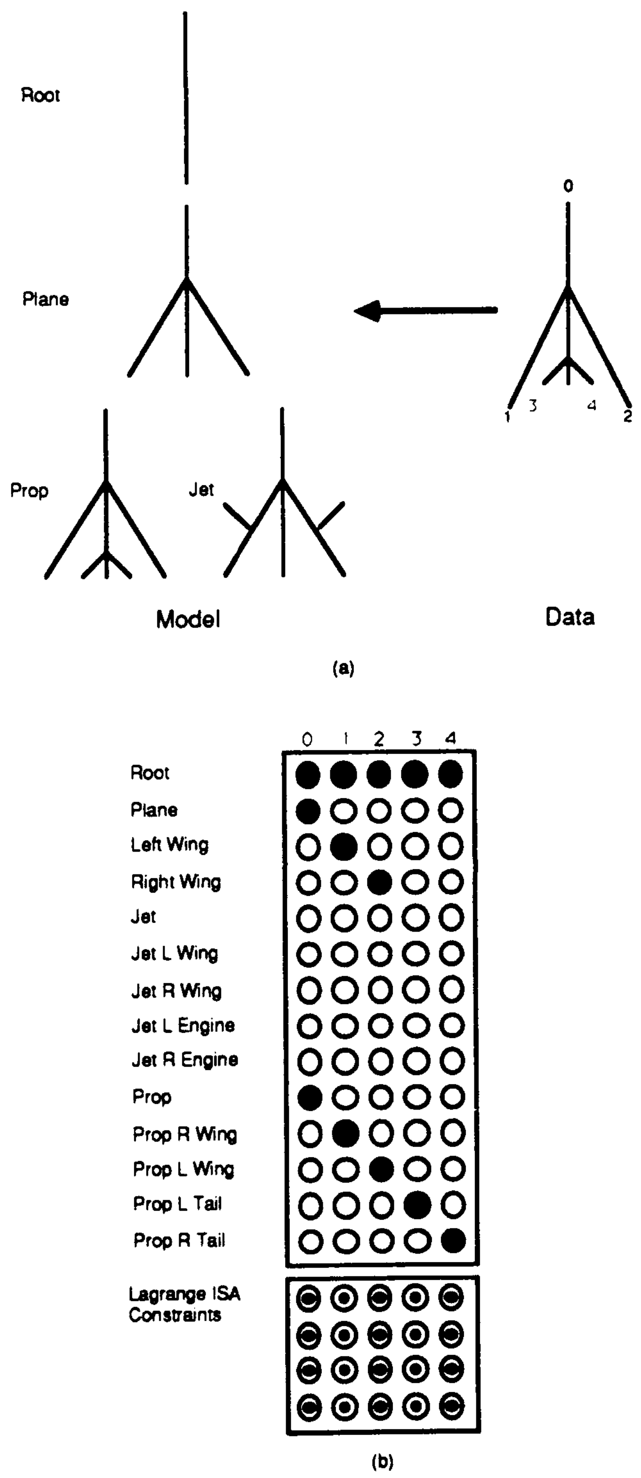


Figure 15: ISA Incremental Match.

The net in (b) has settled on a solution to the graph matching problem in (a). This figure illustrates the use of Lagrange neurons to enforce the *ISA* constraints.

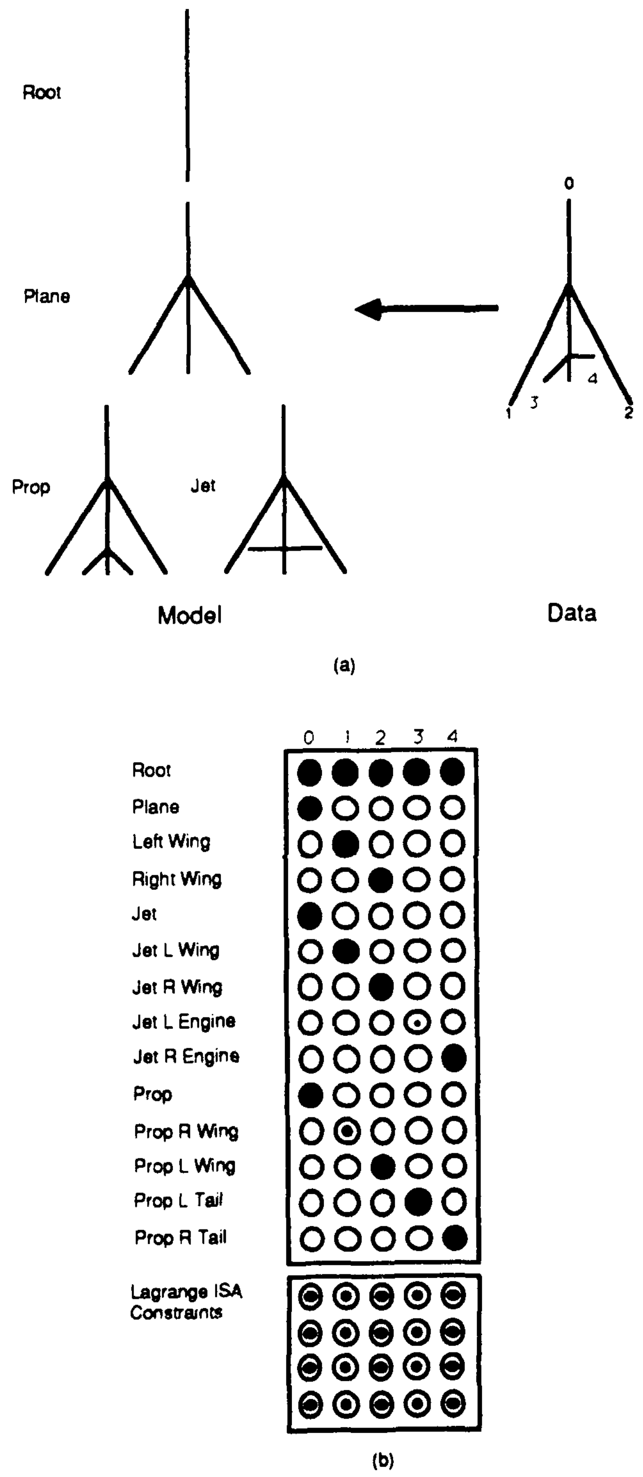


Figure 16: ISA Problem.

The net in (b) has reached a local minima that is not consistent with a valid selection of one of the terminal models. This solution, however, is consistent with the ISA rule, which requires a decision of which part to activate on a local level only. This inconsistency does not occur when the RISA rule is used.

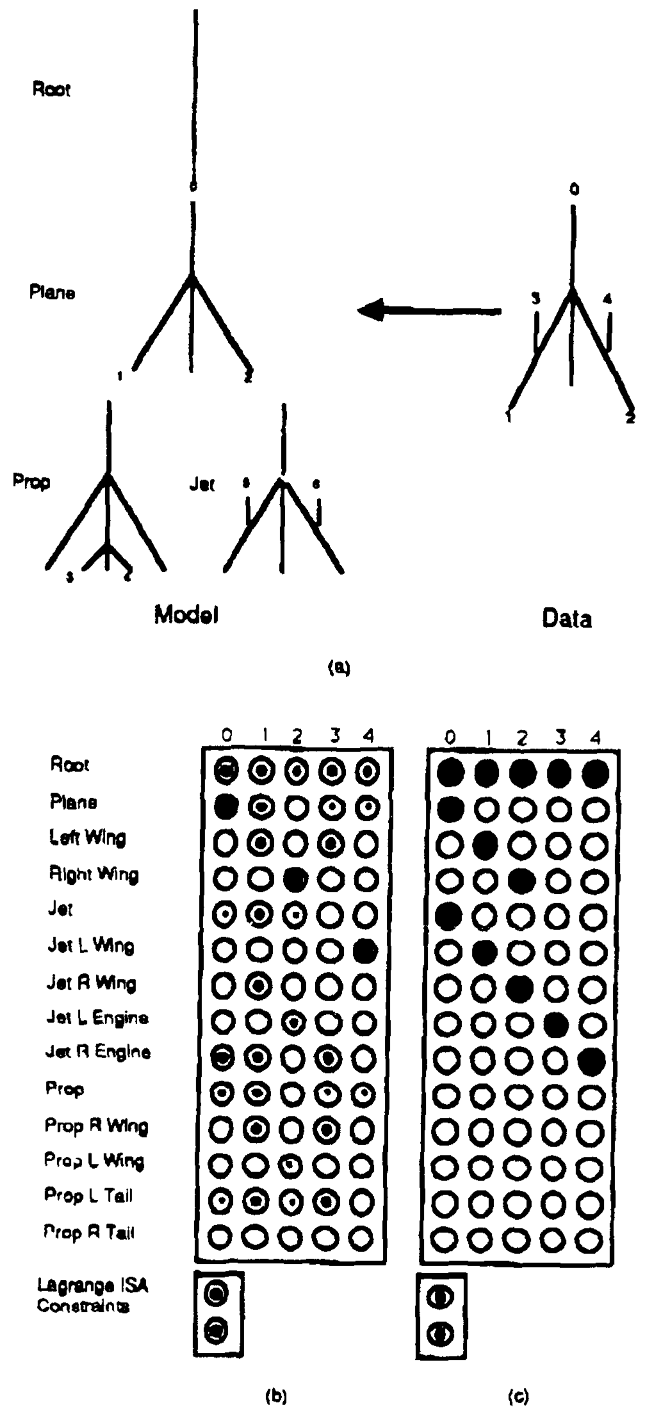


Figure 17: RISA Incremental Match.

The net in (c) has settled on a solution to the graph matching problem in (a). An intermediate point in searching for the solution is illustrated in (b), where appropriate matches are not yet fully activated and several inappropriate matches are active. The *RISA* selection is guided by the unique features of the specialization. This figure also illustrates the use of Lagrange neurons to enforce the *RISA* constraints.

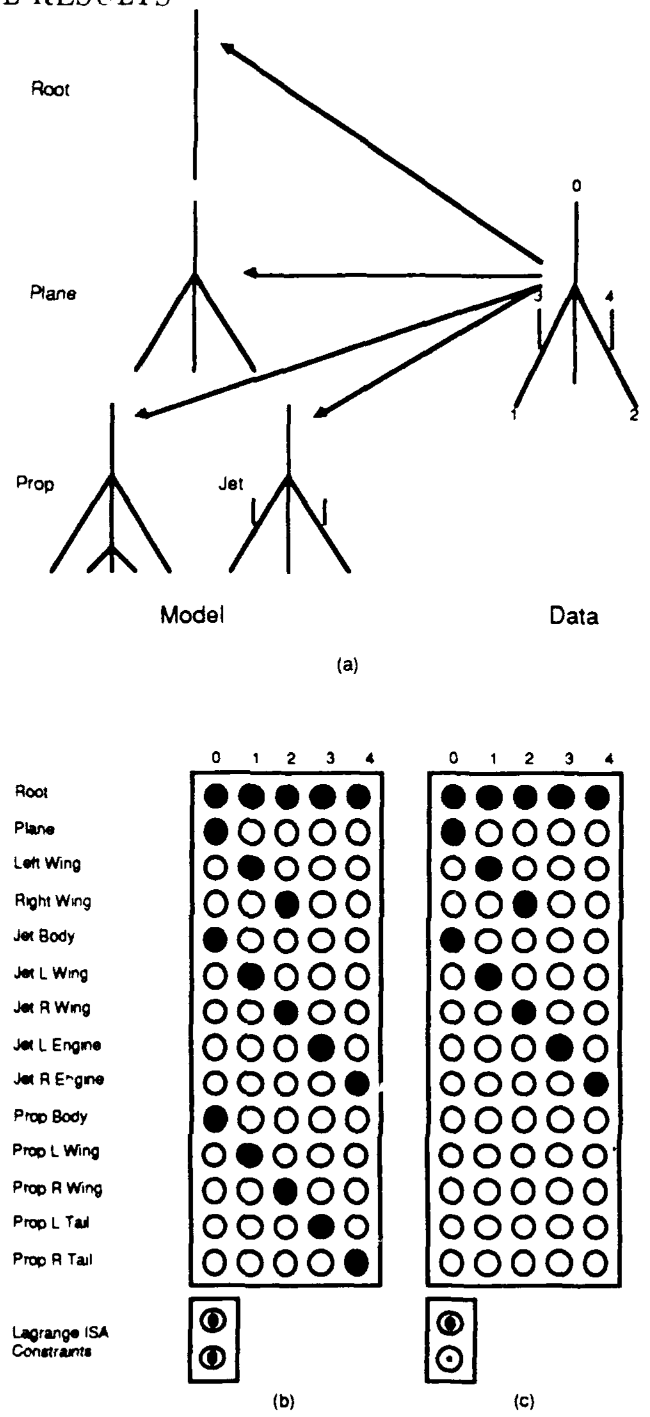


Figure 18: *RISA* hierarchy.

The net in (b), a representative of the graph matching problem illustrated in (a), is started with the data sticks matching to both the *JET* and the *PROP*, a violation of the *RISA* hierarchy rules. In (c), the Lagrange neuron responsible for satisfying the *RISA* constraint exactly increases, eventually forcing the matches to *PROP* parts to disappear.

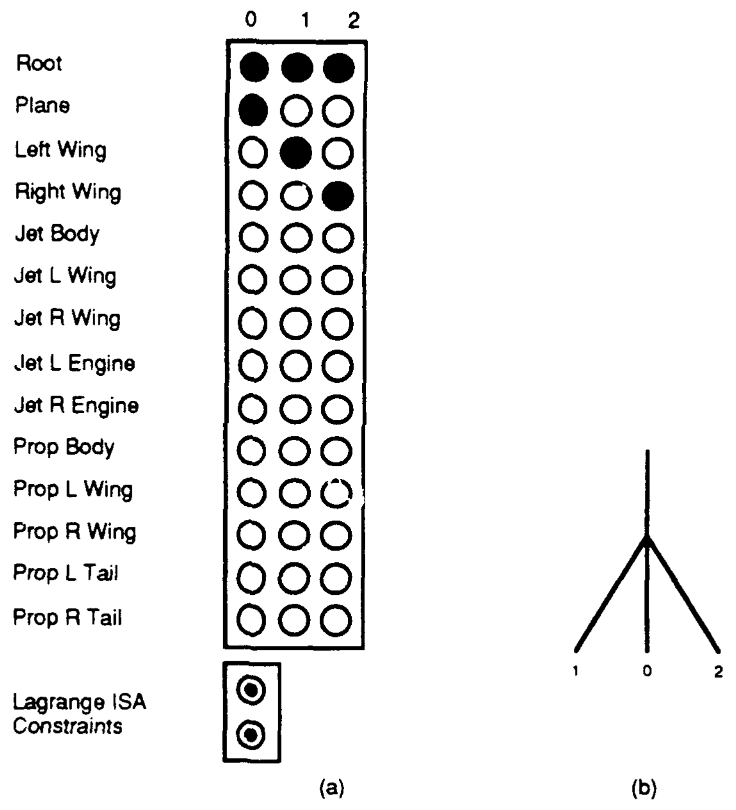
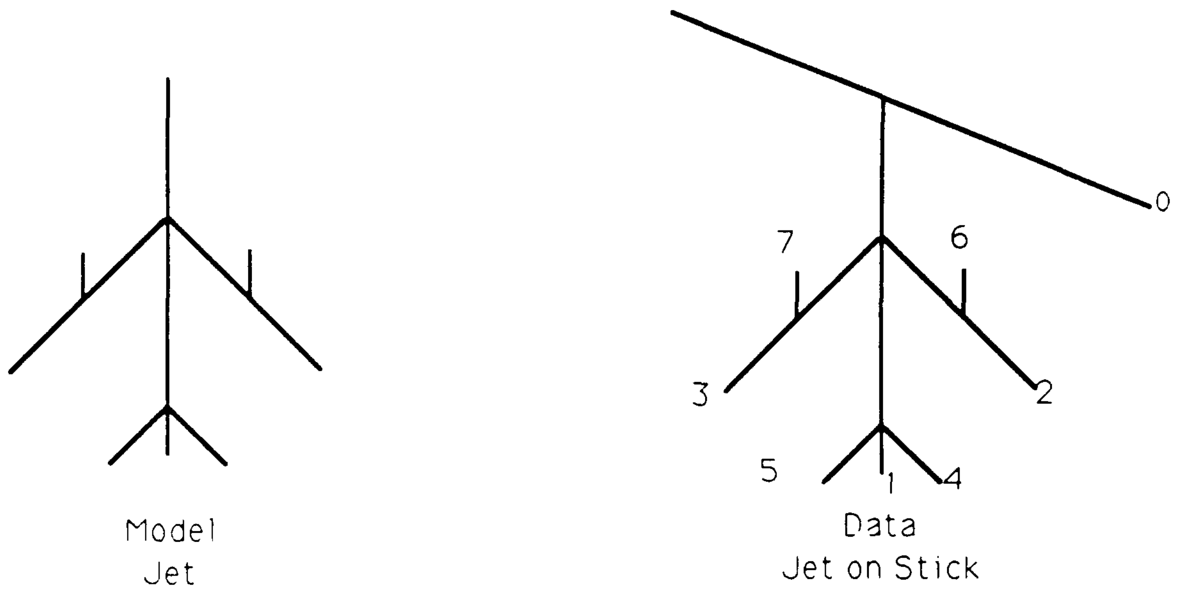


Figure 19: Non-Terminal Match.

It is not necessary to match a terminal model of the *RISA* hierarchy. The net in (a) has matched the data in (b) to *PLANE* only, satisfying the *RISA* hard constraints.



(a)

	0	1	2	3	4	5	6	7
Root	●	●	●	●	●	●	●	●
Jet Body	○	●	○	○	○	○	○	○
Left Wing	○	○	●	○	○	○	○	○
Right Wing	○	○	○	●	○	○	○	○
Jet L Engine	○	○	○	○	●	○	○	○
Jet R Engine	○	○	○	○	○	●	○	○
Prop L Tail	○	○	○	○	○	○	●	○
Prop R Tail	○	○	○	○	○	○	○	●

(b)

Figure 20: Sub-Match.

It is possible to match a subset of a connected assemblage of data sticks to a model. Here, the model *JET* is recognized in a subsection of the data base. There is no specialization hierarchy here.

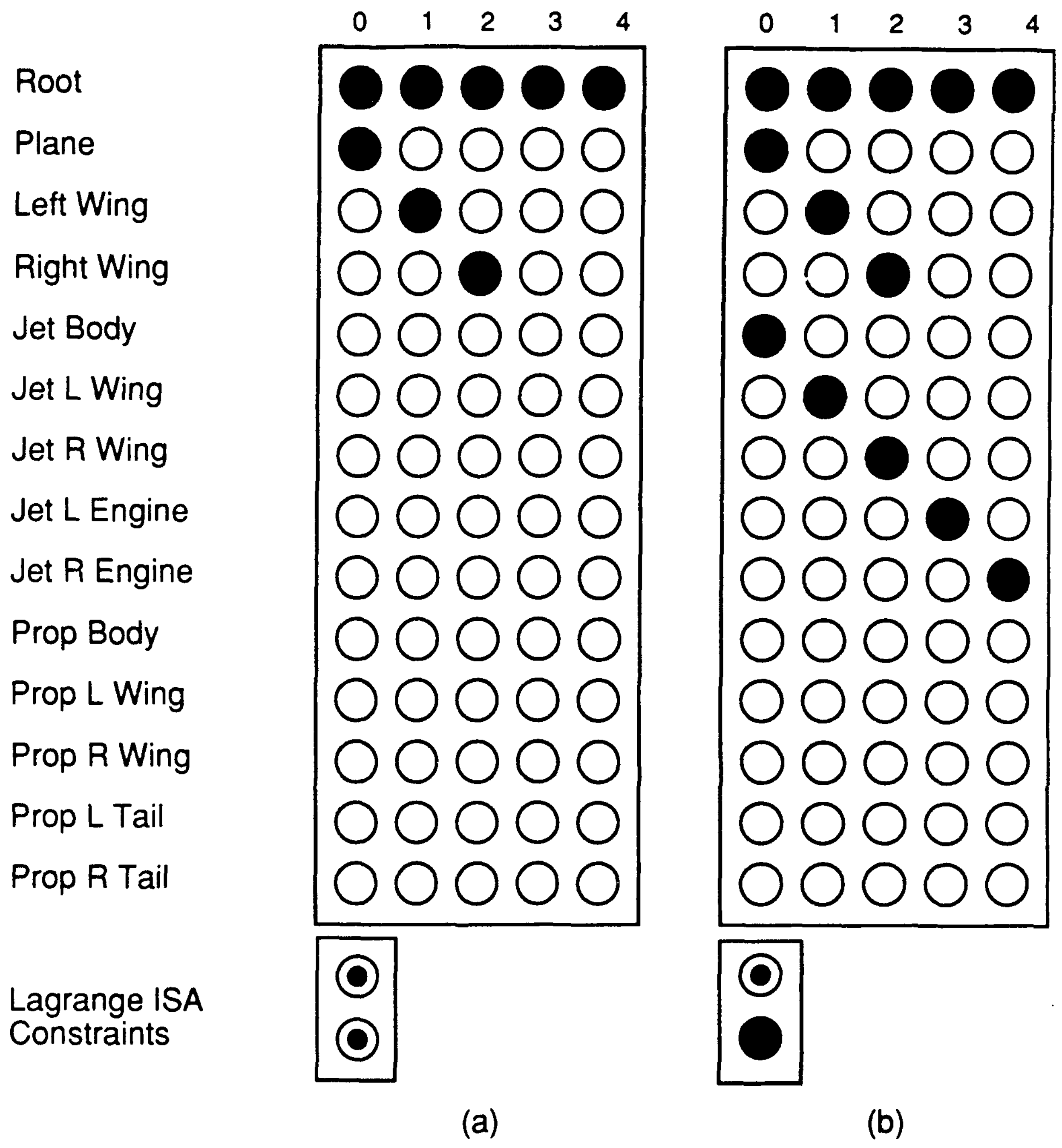


Figure 21: Partial Information.

The matching problem in Figure 18 is started with the M match neurons to *ROOT* and to the subparts of *PLANE* set to active. The network quickly settles on the correct solution of *jet* within 20 time steps, being driven by the *RISA* link from *PLANE*. When a *JET* is matched to *JET* with *ROOT* but not *PLANE* set, not depicted, the network requires 50 time steps to converge to a solution.

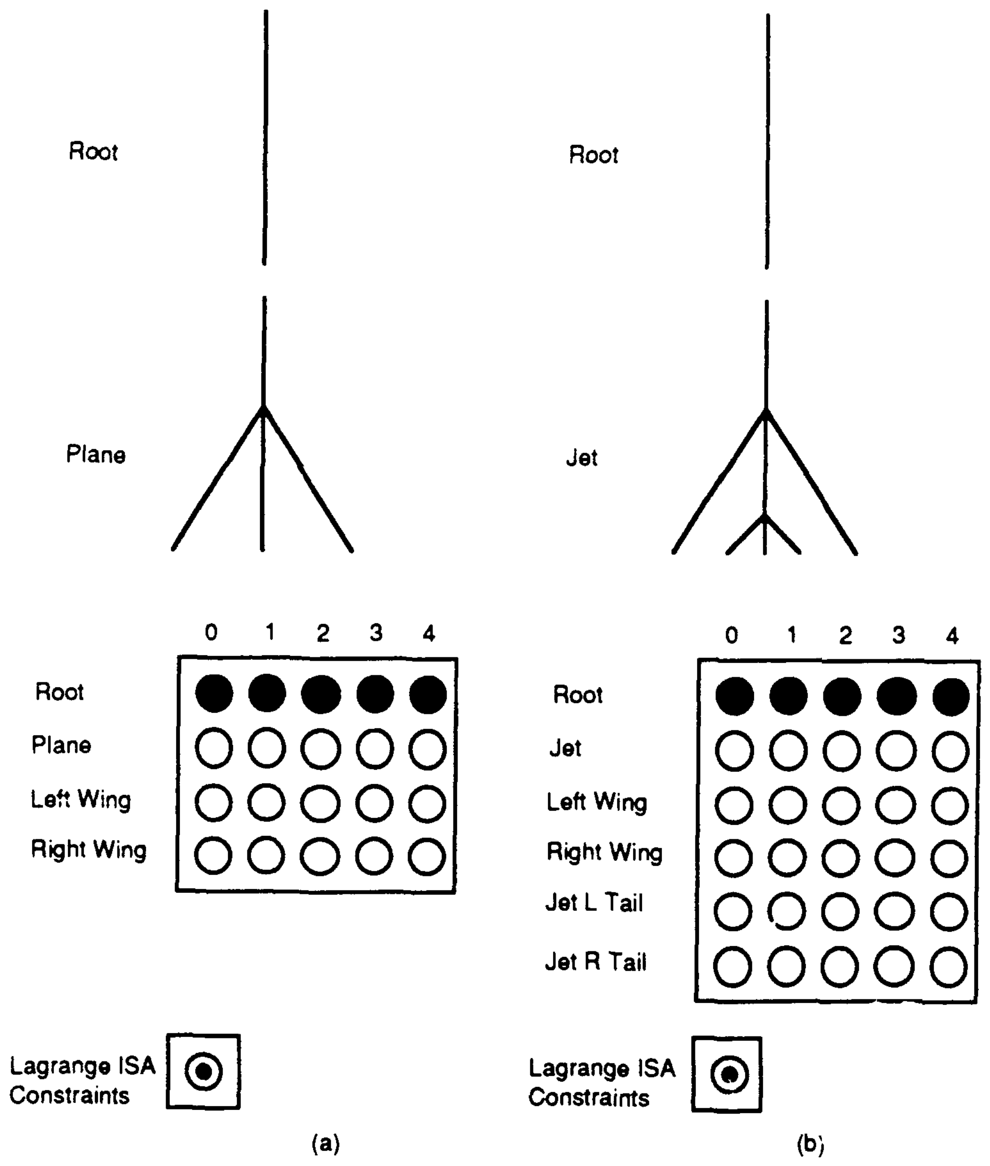


Figure 22: Initial Starting Positions.

Both (a) and (b) show the initial starting positions for the models *PLANE* and *JET*. Both are being matched to a data base containing a *JET*, and are started with the matches to ROOT set to active, Rp and Rj.

10.2 Learning Results

In this section learning of the fit parameters F was implemented. The fixed equation previously employed,

$$F_{fuselage,wing}(\vec{P}_{4,1}) = (1 - ((r_{4,1} - r^{*}_{fuselage,wing})/\sigma_{fuselage,wing})^2)/(\sum_{\beta} INA_{fuselage,\beta})$$

was replaced by a function generated by back-propagation Gaussian neural networks, previously described. Separate neural networks of the form shown in figure 13 were created for each pair (α, β) of model parts to be matched. Each contained from 1 to 25 hidden-layer neurons, each with a set of w, η, λ. The parameters w, η, λ contained initial small random starting values, chosen so that without learning the default output of the network is a near-zero value for any input. This creates a F which discriminates against all matches unless learning has occurred to modify the default response. Learning networks with more hidden-layer neurons allowed more complex learning functions to be generated.

Positive and negative training data of each pair of parts were presented to the training networks, along with the desired output of the learning network. For positive examples, the desired output is 1 (good match), for negative examples, the desired output is 0 (bad match). This process allowed the function $F_{\alpha\beta}$ to be generated specifically for each pair $\alpha\beta$.

In figure 23 (a), a model and two possible training examples are shown. These examples are used to generate the function F shown as the solid line in figure 23 (b). The dotted line in the same figure shows the previously employed fixed F. The difference (1) noted in (b) allowes the matching neural network to discriminate between the good and poor matches in (a) using the learned F, which is more difficult with the smaller difference (2) with the fixed F. The learning function is not limited to generating a F with a single Gaussian-type peak. The complexity of the generated function depends to some extent on the numbers of hidden-layer neurons. An example of this is shown in figure 23 (c), where a F has been generated to match obtuse and sharp angles, while not responding to intermediate angles.

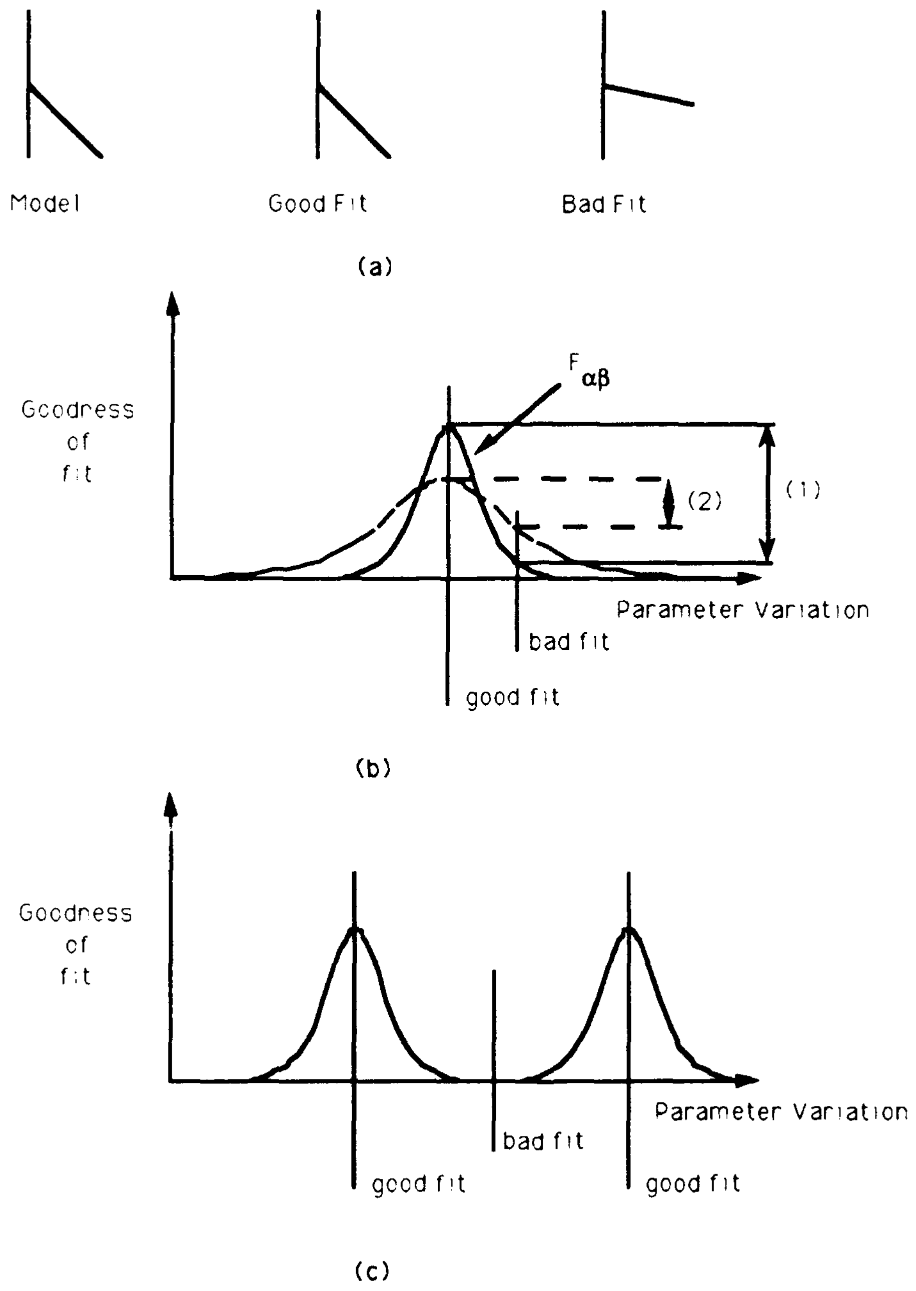


Figure 23: Learning Examples.

11 Discussion

In Stickville, we implement a model-based object-recognition system as a neural network. The network architecture is specified by a third-order objective function, and dynamics derive from standard optimization techniques. The system is "neural" in that a relatively complex calculation is carried out as an analog collective computation of very simple processing elements (neurons) and, at worst, multiplicative connections. The system could presumably be implemented in hardware directly as an analog circuit. (Here we instead simulate the system on an ordinary workstation.)

Other hallmarks of neural nets are learning and fault tolerance. Neither of these features are addressed here, although, as discussed previously, the Stickville net might support the learning of match metrics $F(\vec{P})$. Fault tolerance derives from distributed representations in which information is shared among many neurons. In our case, each "unit" of information, namely $M_{\alpha i}$, is associated with a single neuron in a unary representation. If the neuron or connection to it fails, then the information is lost. Simple tricks like duplicating units can lead to fault tolerance, but distributed representations may have other advantages [11].

Related to our unary representation is the problem of node proliferation. For N sticks and M models and an average fanout of f for INA, ina links, there are NM neurons and $\mathcal{O}(NMf^2)$ wires, clearly a huge number when we expect only $\mathcal{O}(N)$ neurons to be active at the fixed point. It may be possible to greatly reduce the size of the net by transforming the objective function into one which has the same fixpoints but requires less hardware [12].

From the standpoint as a vision system, the task tackled here greatly extends simple recognition strategies, such as template matching, employed in various guises by other neural nets. It falls short in other respects, though. The data organization (*ina* links) are presumed known ahead of time, but a vision system would have to compute these dynamically. (In some real vision applications, this grouping on the data side can indeed be done as a preprocess.) The match metrics $F_{\alpha\beta}(\vec{P_{ij}})$ should also be computed as needed with parameters of the abstraction computed along the way. In fact, objective functions may be specified to do this [7], and experiments were carried out in simple domains that incorporated these features [9], but these achieved limited success. The reasonably successful results for Stickville are encouraging in this light.

A few interesting sidelights are found in Stickville. Because we need to match unequal numbers of parts and models, a novel third-order energy function is formulated to solve this "partial match" problem. In simulations, it works quite well for problems that depend linearly on the data. The use of Lagrange multiplier neurons allowed the introduction of hard constraints within a Hopfield-style network. Such constraints allowed constraints with limited scope (the *ISA* specialization hierarchy) to be held exactly at the solution, while removing the need to select weighting coefficients. The binary constraint rule was implemented as a penalty term in the experiments presented here. It was not implemented as a Lagrange term since each neuron would require its own Lagrange neuron, effectively doubling the size of the network. Strictly speaking it should be implemented as a Lagrange term since the penalty method only reduces the likelihood of selecting a minima within the solution space. This did not appear, however, to be a problem with the results presented here.

One might expect that the hierarchical organization of the database would promote rapid and correct matches. The results with *ISA* hierarchies are so far equivocal, however. The use of an *INA* hierarchy with mainpart abstraction seems to work well as shown in results. For an object of n parts and models, a naive pairwise relational match requires n^4 "rectangles" as opposed to n^2 for the *INA ina* scheme used here. On the other hand, one might expect the use of an *ISA* hierarchy to promote rapid and correct matches. The use of *ISA* links allows partial information about data objects to be stored. Such information then directs further matching, leading to faster convergence. The preliminary results show some advantage in using *ISA* links, but more work is needed.

In summary, Stickville implements in a neural network important aspects of a general recognition strategy as outlined in [7]. Further progress here should be helpful in realizing networks of some consequence for solving real vision problems.

References

[1] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", Proc. Nat. Acad. Sciences USA, vol. 81, pp. 3088-3092, 1984.

[2] J.J Hopfield and D.W. Tank,"Neural Computation of Decisions in Optimization Problems," Biological Cybernetics,**52** (1985)

[3] J.J. Hopfield and D.W. Tank, "Computing with Neural Circuits: A Model," Science **233**, 625 (1986)

[4] G. R. Gindi and D.H. Delorie, "Indexing shapes for Recognition," 5th Intl. Conf on Robot Vision and Sensory Controls, Proc. SPIE,**729**,28-35, Cambridge, Mass (1986)

[5] David G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, (1984)

[6] John Platt and Al Barr, "Constrained differential optimization", Proc. of Neural Information Processing Systems, AIP, Denver, CO 1987, pp. 617-622.

[7] Eric Mjolsness, Gene Gindi, and P. Anandan, "Optimization in Model Matching and Perceptual Organization: A First Look", Research Report YALEU/DCS/RR-634, Yale University, Dept. Computer Science, June 1988.

[8] Eric Mjolsness, Gene Gindi, and P. Anandan, "Optimization in Model Matching and Perceptual Organization", Neural Computation, to appear May 1989.

[9] J. Utans, G. Gindi, E. Mjolsness, and P. Anandan, "Neural Networks for Object Recognition within Compositional Hierarchies: Initial Experiments", Technical Report 8903, Center for Systems Science, Department of Electrical Engineering, Yale University, February 1989.

[10] J.H. Connell, "Learning Shape Descriptions: Generating and Generalizing Models of Visual Objects", MIT AI Lab TR 853, 1985.

[11] David E. Rumelhart and James L. McClelland, *Parallel Distributed Processing*, The MIT Press, (1987).

[12] Eric Mjolsness and Charles Garrett, "Algebraic Transformations of Objective Functions", Research Report YALEU/DCS/RR-686, Yale University, Dept. Computer Science, March 1989.

[13] Tony Zador, Gene Gindi, Eric Mjolsness, and P. Anandan, "Neural Network for Model-Based Recognition: Simulation Results", Neural Networks, vol. 1, Supplement 1, p. 532. (abstract) 1988.

[14] Gene Gindi, Eric Mjolsness and P. Anandan, "Neural Networks for Model Matching and Perceptual Organization", in *Advances in Neural Information Processing Systems I*, p. 618-626, D. Touretzky, ed. Morgan Kaufman, San Mateo, 1989.

[15] J.J. Hopfield and D.W. Tank, "Collective Computation with Continuous Variables", in E. Bienenstock, F. Fogelman Soulie and G. Weisbuck (Eds.), *Disordered Systems and Biological Organization*, NATO *ASI Series*, Vol. F20, pp. 155-170, Berlin, Springen Verlag, 1986.

[16] von der Malsburg, "Pattern Recognition by Labeled Graph Matching", Neural Networks, Vol. 1, p. 141-148, 1988.

[17] D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice Hall, 1982.

[18] D. Marr, *Vision*, W.H. Freeman, 1982.

[19] ,G.H. Shumaker and G. Gindi, "Stickville: A Neural Net for Object Recognition via Graph Matching," Yale University Technical Report No. 8908, April 1989.

[20] C.A. Mead, "Neural Hardware for Vision," Engineering and Science, pp. 2-7, June 1987.

[21] T. Poggio, "Vision by Man and Machine," Scientific American pp. 106-116, April 1984.

[22] H.G. Barrow and J.M. Tenenbaum, "Computational Vision," Proceedings of the IEEE, Vol. 69, No. 5, May 1981, p. 572 - 595.

[23] T.J. Sejnowski and S.R. Lehky, "Neural Network Models of Visual Processing," 1987 Short Course on Computational Neuroscience, Society for Neuroscience.

[24] D.H. Ballard, G.E. Hinton, and T.J. Sejnowski, "Parallel visual computation," Nature, Vol 306, No. 5938, pp. 21-26, 3 November 1983.

[25] C. Kock and S. Ullman, "Selecting One Among the Many: A Simple Network Implementing Shifts in Selective Visual Attention," Massachusetts Institute of Technology, Artificial Intelligence Laboratory C.B.I.P. Paper 003, January 1984.

[26] B. Julesz, "Binocular depth perception of computer generated patterns," Bell Syst. Tech. J. 39, 1125-1162.

[27] R.N. Shepard and J. Metzler, "Mental rotation of three-dimensional objects," Science 171, 701-703.

[28] H.B. Barlow, "Summation and inhibition in the frog's retina," J. Physiol. (Lond.) 119, 69-88.

[29] C.G. Gross, C.E. Rocha-Miranda, and D.B. Bender, "Visual properties of neurons in inferotemporal contex of the macaque," J. Neurophysiol. 35, 96-111.

[30] L.G. Roberts, "Machine perception of three-dimensional solids," in *Optical and electro-optical information processing*, ed. J.T. Tippett et al., 159-197, MIT Press, Cambridge, Mass.

[31] M.B. Clowes, "On seeing things," Artifical Intelligence 2, 1, Spring 1971, pp. 79-116.

[32] D.I. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," Psychology of Computer Vision, 1975.

[33] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," Proc. Roy. Soc. London B. 200, pp. 269-294, 1977.

[34] E. H. Land, "The retinex theory of color vision." Scientific American, December 1977, 108-128.

[35] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," J. Physiol. (Lond) 166, p. 106-152, 1962.

[36] D.E. Rumelhart, J.L. McClelland, "Parallel Distributed Processing Vol I," p. 322-333, 1986.

[37] Moody, "Neural Networks with Gaussian Output Functions," Science 1990.